

AD-A269 782



12

**THE DEVELOPMENT OF AN AUTOMATED COGNITIVE TASK  
ANALYSIS AND MODELING PROCESS FOR INTELLIGENT  
TUTORING SYSTEM DEVELOPMENT**

DTIC  
ELECTE  
SEP 17 1993  
S A D

**KENT E. WILLIAMS  
VIRGINIA TECH**

**FINAL REPORT**

**August 1, 1993**

*Reproduction in whole or in part is permitted for any purpose of the United States Government.*

*This research was sponsored by the Manpower Personnel and Training Program,  
Office of Naval Research under Contract Number N00014-91-J-5-1500.*

*Approved for Public Release: Distribution Unlimited.*

93-21713



13277

# **THE DEVELOPMENT OF AN AUTOMATED COGNITIVE TASK ANALYSIS AND MODELING PROCESS FOR INTELLIGENT TUTORING SYSTEM DEVELOPMENT**

**DTIC QUALITY INSPECTED 4**

**KENT E. WILLIAMS  
VIRGINIA TECH**

**FINAL REPORT**

**August 1, 1993**

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution	
Availability Codes	
Dist	Avail and/or Special
<b>A-1</b>	

*Reproduction in whole or in part is permitted for any purpose of the United States Government.*

*This research was sponsored by the Manpower Personnel and Training Program,  
Office of Naval Research under Contract Number N00014-91-J-1500.*

*Approved for Public Release: Distribution Unlimited.*

REPORT DOCUMENTATION PAGE			Form Approved OMB No 0704-0188	
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</p>				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE August 1, 1993	3. REPORT TYPE AND DATES COVERED Final Report: March 1991-August 1993		
4. TITLE AND SUBTITLE The Development of an Automated Cognitive Task Analysis and Modeling Process for Intelligent Tutoring System Development		5. FUNDING NUMBERS N00014-91-J-5-1500 NR 4428031-01		
6. AUTHOR(S) Kent E. Williams				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Virginia Tech Dept. of Industrial and Systems Engineering - 1900 Kraft Drive Blacksburg, VA 24060		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of the Navy S.O. Code 342CS Office of Naval Research 800 North Quincy Street Arlington, VA 22217-5660		10. SPONSORING / MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release: Distribution Unlimited			12b. DISTRIBUTION CODE	
<p>13. ABSTRACT (Maximum 200 words)</p> <p>Studies have shown that 98% of students with private tutors do better than students in the traditional classroom. This has initiated research and the development of computerized intelligent tutoring systems based upon research related to cognitive learning and the communication of knowledge between a tutor and a student. Intelligent tutoring systems have demonstrated a 65% improvement in what can be learned in a given period of time as compared to traditional computer aided instruction in Navy tasks. The principle component of these computerized tutors is a model of the knowledge which makes up the cognitive task to be learned by a student. Creating these cognitive models requires an extensive analysis of the task knowledge. This process is expensive yet critical to the success of these tutors. This research has produced a software tool which interacts with a subject matter expert and conducts an analysis of his/her knowledge, building the needed cognitive task model for use in intelligent tutoring system development. An evaluation of the system was conducted to determine the degree of variation in models generated by experts on a specific highly proceduralized task. The subject generated models were also</p>				
14. SUBJECT TERMS Cognitive Task Analysis, Cognitive Simulation Modeling Intelligent Tutoring Systems, GOMS, Human Systems Interaction Transfer of Training, Training Time Prediction			15. NUMBER OF PAGES 119	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited	

compared to a model generated by a cognitive analyst. The results demonstrated a high degree of consistency in the models developed between subjects although subjects tended to organize their knowledge of procedures into units of differing size. The subject models as compared to that model generated by a cognitive analyst, showed considerable agreement except that subject matter experts tended not to describe mental operations involved in the task as frequently as did the cognitive analyst. Such mental operator descriptions are critical to developing cognitive task models for predicting the execution time of a procedure when creating models of human computer interaction and when predicting training time from these models. On the other hand, many primitive mental operations such as adding and deleting information in working memory or retrieving and storing information in long-term memory are abstractions of the cognitive analyst. These operations would not occur in task descriptions of curriculum except for curriculum in cognition. Therefore, the absence of description of mental operators on the part of subject matter experts when explicating knowledge about their task does not limit the aids applicability to the generation of task knowledge to be used for developing curriculum content.

Although the design of the system was targeted for the acquisition and modeling of procedural knowledge describing how to use a specific device or system, it can readily be applied for developing qualitative models of how a system works and for structuring knowledge about concepts, their attributes and relations between concepts and attributes.

Numerous applications and benefits which can be achieved from this cognitive task modeling capability as they relate to human systems interaction, training systems development, transfer of training, and job design are discussed.

## TABLE OF CONTENTS

	<u>Page</u>
Introduction .....	1
Production System Framework.....	2
Applicability of Cognitive Models.....	3
Intelligent Tutoring Application.....	3
Predicting Human Performance .....	4
Problem Solving.....	5
Human Learning.....	6
Tactical Decision Making Training .....	7
Objective of Research.....	8
Design Guidelines.....	9
Review of Methodologies, Results and Conclusion.....	10
Opening a File.....	21
Specifying the Main Goal .....	21
Creating a New Method .....	22
Decision, Store, Recall, and Go To Steps .....	24
Ordering Steps.....	28
Creating Alternative Methods.....	29
Creating Selection Rules.....	31
Quitting the Process .....	34
Creating Exception Rules.....	35
Returning to Model After Completion of Exception Rule Goal .....	38
Impass Resolution/Overcoming Failures .....	41
Navigating and Editing A Model .....	46
Making Changes to a Model of Editing.....	53
Changing Step Order.....	57
Adding a New Method to a Model .....	58
Adding a Step in a Method.....	59
Consolidating Steps into a New Method.....	59
Graphical Editing .....	60
Executing a Model .....	61
Formative Evaluation of Modeling Capability .....	79
Methodology.....	80
Subjects .....	80
Experimental Materials and Equipment.....	80
Experimental Procedure .....	80
Experimental Analysis .....	92
Results.....	93
ANOVA on Accuracy Measures.....	95
Analysis of Consistency.....	97
Discussion and Conclusion .....	101

## TABLE OF CONTENTS

	<u>Page</u>
Implications for HCI Complexity Predictions.....	103
Predicting Time to Learn and Degree of Training Transfer for Extending CAT .....	105
Practical Benefits Relative to Manpower, Personnel and Training.....	115
Acknowledgment .....	116
References .....	117

## LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
Figure 1 - Taxonomy of Knowledge Acquisition Methods .....	13
Figure 2 - An example of a GOMS hierarchy .....	14
Figure 3b - Logic flow diagram for impass resolution subsystem in CAT .....	18
Figure 3c - Logic flow diagram for step interpretation subsystem in CAT .....	19
Figure 3d - Logic flow diagram for selection rule subsystem in CAT .....	20
Figure 4 - Open File Dialog .....	21
Figure 5 - Main Goal Dialog .....	22
Figure 6 - New Method Dialog .....	23
Figure 8 - Decision Step Dialog .....	24
Figure 9 - Query Store Dialog .....	25
Figure 10 - Store Step Dialog .....	26
Figure 11 - Query Recall Dialog .....	27
Figure 12 - Recall Step Dialog .....	27
Figure 13 - Step Order Dialog .....	29
Figure 14 - Grouping Steps .....	29
Figure 15 - Consolidation Help Dialog .....	30
Figure 16 - Any More Methods Dialog .....	31
Figure 17 Selection Rule Dialog .....	32
Figure 18 - Another Selection Rule Dialog .....	33
Figure 19 - Describe Methods Dialog .....	34
Figure 20 - Introduction to Exception Rules Dialog .....	35
Figure 21 - Add Exception Rule Dialog .....	36
Figure 22 - Create Exception Dialogue .....	37
Figure 23 - Exception Return Dialog .....	38
Figure 24 - Restart Goal Dialog .....	39
Figure 25 - Introduction to Impass Resolution Dialog .....	41
Figure 26 - Primitives Which Can Fail Dialog .....	42
Figure 27 - Impass Resolution Dialog .....	43
Figure 28 - Model Definition Complete Dialog .....	45
Figure 29 - The Navigate Menu .....	46
Figure 30 - View Goal Dialog .....	48
Figure 31 - View Goal Containing Dialog .....	49
Figure 32A - Logic flow diagram for View Goal Menu Commands in CAT .....	50
Figure 32b - Logic flow diagram for Selection Rule and Exception Rule Menu Commands in CAT .....	51
Figure 32c - Logic in flow diagram for Editing and Quitting Menu Commands in CAT .....	52
Figure 33 - Goal Edit Dialog .....	54
Figure 34 - Condition Edit Dialog .....	54

## LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
Figure 35 - Method Edit Dialog.....	55
Figure 36 - Step Edit Dialog.....	55
Figure 37 - Selection Rule List Dialog.....	56
Figure 38 - Exception Rule List Dialog.....	57
Figure 39 - Graphic Editing.....	61
Figure 40a - Logic flow diagram for Model Execution process in CAT.....	63
Figure 40b - Logic flow diagram for Goal Execution process in CAT.....	64
Figure 40c - Logic flow diagram for Method Execution process in CAT.....	65
Figure 40d - Logic flow diagram for Step Execution process in CA.....	66
Figure 40e - Logic flow diagram for Exception Execution process in CAT.....	67
Figure 41 - Execute Goal Dialog.....	68
Figure 42 - Selection Rules Introduction Dialog.....	69
Figure 43 - Condition Selection Dialog.....	69
Figure 44 - Goal Failed Dialog.....	71
Figure 45 - Execute Method Dialog.....	71
Figure 46 - Method Failed Dialog.....	72
Figure 47 - Goal Accomplished Dialog.....	72
Figure 48 - Step Condition Dialog.....	73
Figure 49 - Execute Primitive Dialog.....	74
Figure 50 - Query Execute Primitive Dialog.....	75
Figure 51 - Execution Exception Dialog.....	77
Figure 52 - Execute Exception Rule Dialog.....	78
Figure 53 - Exception Rule Accomplished Dialog.....	78
Figure 54 - Prepare to drive a car demonstration model.....	84
Figure 55 - Mail a letter demonstration model.....	85
Figure 56 - Model of baseline knowledge for the experimental task.....	86
Figure 57 - Interaction of Method of Type by Primitive Step Type as Percent Accuracy.....	96
Figure 58 - Examples of rules showing transfer status.....	110
Figure 59 - Comparison of an expert level representation of a method to a novice level representation of a method.....	113

## LIST OF TABLES

<b><u>Table</u></b>	<b><u>Page</u></b>
1 Task Explanation for Screening Session.....	82
2 Task Explanation for Machine-Aided Session.....	87
3 Example Calculations for the Accuracy Performance Measure.....	90
4 Example Calculations for the Consistency Performance Measure.....	91
5 Summary of Consistency Results of Subject-Versus-Subject Comparison.....	98
6 Confidence Intervals for Performance Measures.....	100

## Introduction

The objective of this project was to design, develop and formatively evaluate a computerized tool to aid in the conduct of a detailed cognitive task analysis. A cognitive task analysis provides an explicit description of the procedural knowledge required to perform a task. These descriptions are referred to as cognitive models or mental models. These models consist of a set of procedural knowledge units which can be executed by a production system to simulate how to do something. Consequently, the kinds of tasks which can be modeled as a result of this research are limited by two restrictions. First, the tasks must specify how to do something as opposed to how something works. Kieras and Polson (1985) have discriminated between these two types of knowledge by referring to them as "how-to-do-it" knowledge and "how-it-works" knowledge, respectively. "How-to-do-it" knowledge refers to that knowledge which explicitly describes, for example, how to drive a car. "How-it-works" knowledge explicitly describes the inner workings of the physical system which enables a car to be driven. It appears from our research that the process of eliciting knowledge which explicitly describes "how-to-do-it" versus "how-it-works" is the same; however, the instructions for eliciting "how-it-works" knowledge requires different semantics. Likewise, the knowledge units for mentally modeling how to do something and how something works appear to share the same structure.

The second constraint limiting the range of tasks which can be modeled as a result of this research effort, was that the units of knowledge making up the model conform to the structure of a production unit executable by a production system architecture of cognition. Production systems of cognitive psychological processes were introduced by Newell (1973) and Newell and Simon (1972) as a result of their work in understanding human problem solving. Production systems were later applied to human learning by

Anderson Klein and Beasley (1980). A production system was also the fundamental architecture for Anderson's (1983) Adaptive Control of Thought (ACT) model. Since then the production system architecture has been applied to modeling a wide range of cognitive tasks (Klahr, Langley and Neeches, 1987).

### **Production System Framework**

A production system consists of a collection of production rules and a working memory. The working memory consists of a representation of the current goal(s) which the system is attempting to achieve, the status of current and past actions, and information about the current environment. A production rule takes the form of a condition-action pair: IF (condition(s) THEN action(s)). The condition side of the rule consists of the contents of working memory: specific goals, the status of recent past and current actions, and specific input from the environment. The action(s) can consist of overt behavioral acts or the addition or deletion of information in working memory. A whole set of these production rules make up a program which, when executed, simulates how to do something.

The process which drives this simulation is referred to as the recognize-act process. During the recognize stage, the condition side of the rules making up the program are compared to the contents of working memory. If a rule's conditions match with the contents of working memory then the action side of the rule is fired. This action in turn changes the contents of working memory. On the next cycle of the process, the recognize stage again compares the new contents of working memory with the set of production rule conditions and fires another action if the conditions of a rule match. This process continues until the goal which motivates the system is achieved. Therefore, given

this definition, the objective of conducting a cognitive task analysis is to create an explicit set of production rules having the structure: IF (condition(s) THEN action(s)). This knowledge program can be executed by a production system to simulate the mental and behavioral operations required to perform a task. The resultant models created by such an analysis have been demonstrated to be extremely accurate in predicting human performance on procedural tasks (Card, Moran and Newell, 1983; Kieras and Polson, 1985; Polson, 1987; Kieras and Meyer, 1992; Gray, John, and Atwood, 1992 and Peck and John, 1992).

### **Applicability of Cognitive Models**

To date, the cognitive models which result from a detailed cognitive task analysis have been applied to the development of intelligent tutoring systems, the prediction of human performance, the evaluation of the complexity of human computer interfaces, the development of models of human problem solving, the understanding of human learning, and the development of real time knowledge-based simulations of tactical decisions.

### **Intelligent Tutoring Application**

The intelligent tutoring system case employs a cognitive task model of some domain which is used to monitor the student's performance and to diagnose difficulties which the student is having in solving domain related problems. A notable example of an intelligent tutor employing a cognitive task model is Anderson's LISP tutor (Anderson, Conrad and Corbett, 1989 and Corbett and Anderson, 1989). As Anderson et al (1989) have pointed out, one of the important benefits of this tutoring work is the careful task analysis that goes into developing the production system model of the domain task. This

analysis yields a more systematic and critical development of the curriculum for instruction. Indeed Williams, Reynolds, Carolan, Anglin and Shrestha (1990) demonstrated a 65 percent improvement in skill acquisition by restructuring existing curriculum employing a detailed cognitive task analysis approach.

### **Predicting Human Performance**

The prediction of human cognitive performance is another case in which a detailed cognitive simulation model is a fundamental requirement (Card, Moran and Newell, 1983; Kieras and Polson, 1985; Kieras, 1988; Gray, John and Atwood, 1992 and John, 1990). Recently Kieras and Meyer (1992) have demonstrated a 99 percent agreement between observed behavior and predicted behavior when predicted performance was generated employing a detailed cognitive task model of multi-task performance. Having the power to predict performance via a computational model of a cognitive task can have numerous applications. One can accurately predict human performance in hypothetical situations prior to the costly design and development of prototype systems which must be subjected to real world tests. The capability to predict performance of a human operator while conducting a task in a complex man-machine environment can reduce the time to develop systems. This prediction can ensure that the human can perform the tasks given a proposed system design, such that the benefits of a proposed system can be fully achieved. Human performance prediction is especially important in the design of man-machine interactions. The ability to predict human operator performance by cognitive simulation modeling can identify potential design features which may inherently generate operator error due to exceeding human capacities to process information. Predicting human performance by modeling the cognitive tasks which describe man-machine interactions can therefore greatly reduce design, development and evaluation costs.

## **Problem Solving**

From a research perspective, cognitive task modeling is fundamental to explanations of human problem solving and human learning. The early work of Newell and Simon (1972) focusing upon the discovery of the underlying processes of problem solving, emphasized the importance of a problem representation in the problem solving process. In accordance with the formulation of the problem solving process, there are two stages to problem solving: the representation stage and the solution stage (Simon, 1978). The representation stage consists of describing the goals, subgoals, the initial conditions and the various alternative operations which can be applied to achieve different specific subgoals in an attempt to achieve the end goal. In short, the representation stage consists of the development of a cognitive model. All of the relevant problem knowledge in the form of IF (goal, subgoal and conditions) THEN (action) must be retrieved from memory and integrated in such a fashion as to create this problem representation. The representation or mental model of the problem to be solved is fundamental to the problem solving process. Without an adequate cognitive model of the problem, the solution cannot be achieved. The solution stage of problem solving consists of making the right choice between alternative actions to apply to the problem in order to reach specific subgoals, and eventually, the end goal. This implies that good problem solvers have comprehensive representations or cognitive models made up of many knowledge units of the IF-THEN form which can be integrated and assessed in terms of their applicability in attaining intermediate subgoals and the end state. Consequently, the ability to generate comprehensive cognitive models of a problem situation is fundamental to problem solving. This formulation of problem solving has been shown to be applicable to such important activities as scientific discovery. The work of Langley, Simon, Bradshaw and Zytkow (1987) and Qin and Simon (1990) has demonstrated that the processes which describe and

replicate how some of the great discoveries in science were achieved, such as Kepler's laws of motion of planetary bodies, Boyle's law of behavior of gases under pressure, Galileo's law of uniform acceleration, and Ohm's law of electrical conductivity, are in fact no different than those processes observed in all kinds of problem solving situations.

### **Human Learning**

Other research evolving from studies in problem solving has focused upon an investigation of what is referred to as the "self explanation effect". The self explanation effect is used to describe why good reasoners or learners differ from poor learners even when good and poor learners are approximately equal in intelligence and prior knowledge about a domain. Apparently, good learners generate more explanations for the actions they take in developing a solution to a problem. Poor learners on the other hand do not generate sufficient explanations relative to identifying conditions associated with actions taken (Chi, Bassok, Lewis, Reimann and Glaser, 1989). In an attempt to determine exactly how these good learners do what they do, researchers (Van Lehn, Jones and Chi, 1991) have built production system models to determine differences in the way in which good and poor learners represent and solve problems. The cognitive model building process is crucial to the conduct of such learning research.

Other learning research conducted by Kieras and Bovair (1986) and Bovair, Kieras and Polson (1990) related to the case of learning a cognitive task has demonstrated that the time to learn a text editing skill can be predicted from a detailed cognitive task model. Consequently, as we again see, cognitive modeling is essential for the conduct of research in such important areas as human learning and problem solving.

## **Tactical Decision Making Training**

As a final point in motivating the significance of cognitive task modeling, it is important to note the applicability of cognitive task modeling for simulator based training systems. Tactical decision making can be extremely complex depending upon the weapon, sensor, countermeasure and maneuvering characteristics of the targets which are engaged in the tactical situation. Consequently, training individuals to make the right decisions in deploying their systems in a tactical situation is, likewise, complex. An important aspect in training tacticians is to provide a simulation environment in which tacticians can acquire knowledge of enemy tactics and apply their knowledge of tactics in making decisions to overcome the enemy. Of major importance in tactical decision making training is knowing what the enemy can do. Knowing what the enemy can do allows the tactician to formulate hypotheses about potential enemy action, verify a specific hypothesis about enemy intent, and take action to both counter enemy actions and effectively eliminate the threat. Tactical training, therefore, requires an intimate knowledge of enemy tactics. This knowledge is typically acquired in the classroom and then applied in simulator based training scenarios. The dynamic exercises generated by simulator based training systems must therefore demonstrate, with a high degree of fidelity, the tactical behavior of enemy targets. Williams and Reynolds (1990) developed, demonstrated, and evaluated the fidelity of a number of Soviet platforms (i.e. air, surface, and subsurface) the tactical behavior of which was controlled by cognitive models of Soviet threat tactics in a dynamic training simulator environment. Additionally, Williams and Reynolds (1990) provided simulations of other U.S. Naval platforms operating in consort with each other in a combined arms operation against hostile targets. Expert Naval tacticians agreed that the tactical behavior of the simulated targets were within 80 percent of what they would expect in the real world if they were faced with a similar tactical situation.

The benefit of such simulation for training purposes is a reduction in the manpower required to maneuver targets during simulation exercises. (Targets are typically manually maneuvered by problem controllers during the conduct of simulation exercises). Furthermore, these cognitive model controlled simulations provide greater realism (i.e. functional fidelity) to the training situation. Tacticians who can acquire the cognitive models of hostile tacticians are better prepared to formulate hypotheses and reason about hostile intent. Solving problems about adversaries, therefore, requires that the tactician has a comprehensive model of not only his own tactical capability but also that of his opponent (Thagard, 1992). Cognitive model based simulations of opponent tactics can be quite useful in mission planning and assessing the intent of the opponent during a dynamic engagement.

### **Objective of Research**

Given the breadth of applications which have involved production system models of cognitive tasks, and the impact of such modeling in terms of training gain and the ability to predict human computer interaction, it appears that this modeling process is quickly becoming a core technology for advanced systems development. However, the benefits from the application of this core technology will not become fully realized until the cost to develop such models is reduced. Presently the cost of cognitive task modeling is driven by the scarcity of those individuals skilled in cognitive modeling and the intensity of labor required to create such models. Consequently, the objective of this research was to identify and systematize a process for the conduct of a cognitive task analysis which would lend itself to automation as an interactive computerized aid. The initial application for an automated process was targeted for capturing knowledge about "how-to-do" something

which in turn would be used in developing ideal student models for production system based intelligent tutors for Navy systems operator training.

### **Design Guidelines**

In keeping with the objective of this research, a number of design guidelines were developed to aid in identifying and selecting a cognitive task modeling methodology which would be applicable to the initial target domain of ideal student modeling for procedural tasks, as in device operations. The design guidelines consisted of five constraints on the chosen methodology.

The first constraint addresses the nature of the production system architecture of cognition. The methodology must yield units of knowledge whose structure and form is consistent with the condition-action units typical of production systems, since this form of knowledge representation has been shown to mimic human cognition.

The second constraint refers to the breadth of task domains to which such a system can be applied without the need of prior knowledge about a domain. This is a general purpose criterion. Some cognitive modeling or knowledge acquisition techniques require interaction with an individual or a computerized aid which shares some degree of knowledge about the task being modeled. Therefore, some automated acquisition techniques require preprogramming a portion of knowledge about a domain prior to eliciting more detailed knowledge from an expert. This process would obviously lend additional cost to automating the modeling process and would reduce user acceptability of such an automated aid. The objective is to select a methodology that does not require prior task knowledge.

The third constraint is that the methodology must involve a highly systematic process. The process must be specified to a level of detail which lends itself to translation into a formal computer program.

The fourth constraint addresses ease of use. The methodology must elicit knowledge from a user in a manner which is simple and natural, and the interface must guide the user through the process without requiring the user to learn a large number of commands. The process must be compatible with the way in which people typically describe procedural tasks.

The fifth and last constraint refers to the validity of the methodology. The methodology must generate models of cognitive tasks which have been empirically demonstrated to be predictive of human performance. This last constraint lends credibility to the process in that the models can be used to make predictions about human performance which correlates highly with observed behavior.

### **Review of Methodologies, Results and Conclusions**

The search for an appropriate methodology led to a review of knowledge acquisition techniques for eliciting and/or acquiring representations of cognitive structures from experts. Approximately fifty different sources of knowledge acquisition methodologies were reviewed. From this review, a classification scheme for organizing the methodologies was developed. A taxonomy of knowledge acquisition techniques was generated from this initial classification. Instances of techniques which were referred to by different names, but still shared fundamentally similar processes and purposes, were grouped together and a simple instance was selected from this group to represent the methodology in the taxonomy. The high level classification grouped the methodologies

along a continuum describing the degree which automation played in the elicitation process. The resultant classes (See Figure 1) consisted of manual methods which included little if any automation, machine-aided methods which involved interaction between an expert and a computerized aid, and machine learning methods which required minimal human interaction (Williams and Kotnour, 1993).

Each of the methodologies included in the taxonomy was then assessed against the design guidelines. Those methodologies which were not consistent with each and every constraint of the guidelines were eliminated from further consideration. As a result of this process of elimination, two methodologies were found to meet each constraint specified in the guidelines. These two methods fall within the manual methods techniques and have been referred to as "cognitive task analysis" methods.

The two methods identified were GOMS (Card, Moran and Newell, 1983) and Constructive Interaction (Miyake, 1986). Each method employs a top-down, breadth-first decomposition process. In a GOMS analysis, the decomposition operates on goals to be accomplished by steps of a procedure. In the Constructive Interaction case, the decomposition operates on functions to be realized by a set of mechanisms. The difference between these two methods lies in the type of knowledge they model. Constructive Interaction describes a process employed by individuals communicating their understanding of how a complex physical device works. That is, the result of the process yields "how-it-works" knowledge. The GOMS technique on the other hand was designed to yield "how-to-do-it" knowledge units. Each method builds a hierarchy of knowledge. The Constructive Interaction process yields a hierarchy of functions and mechanisms to produce these functions, whereas the GOMS process yields a hierarchy of goals and methods to accomplish those goals. Since the GOMS analysis process has been well

# KNOWLEDGE ACQUISITION METHODS

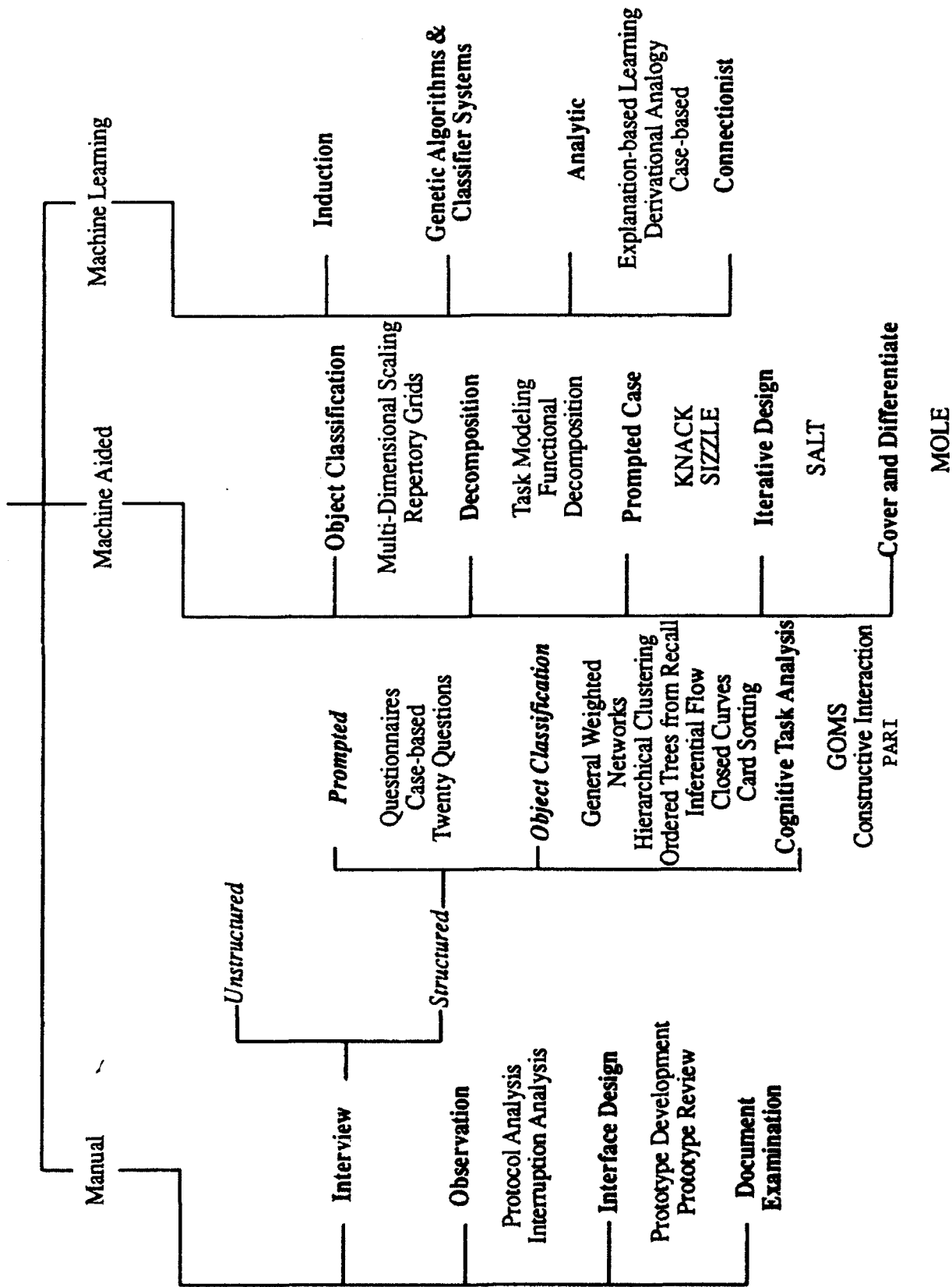


Figure 1 - Taxonomy of Knowledge Acquisition Methods

developed for practical application by Kieras (1988 and 1991) the GOMS methodology was selected after implementation as a computerized structured interview aid to develop cognitive models of tasks.

### **GOMS Analysis Process**

The GOMS process involves specifying four types of components to a cognitive task. These components are (1) goals - what needs to be accomplished, (2) operators - the steps which must be taken to accomplish a goal, (3) methods - a collection of steps which must be executed in some specified order to accomplish a goal and (4) selection rules - a set of conditions which determines which method to select if more than one method can be executed to achieve a specific goal. The relationship of these components is presented in Figure 2 as a goal-method hierarchy.

The GOMS process is initialized by specifying a top level goal to be accomplished. Next, a set of high level steps making up a high level method must be identified to accomplish the high level goal. Typically there is only one high level method for accomplishing the top level goal at this stage of the analysis. Each step in this high level method is then made a subgoal, which in turn can be accomplished by another method or set of methods. This is what is meant by top-down decomposition. In left to right order, each new subgoal must then be assigned a method or set of methods which can accomplish the subgoal. If more than one method can be used to accomplish a specific subgoal, then a selection rule must be defined which specifies under what conditions one would execute one method as opposed to some other method. When methods have been identified for the first subgoal, the next subgoal in the tree is addressed in order to specify the method or methods appropriate for accomplishing it. When methods for each subgoal

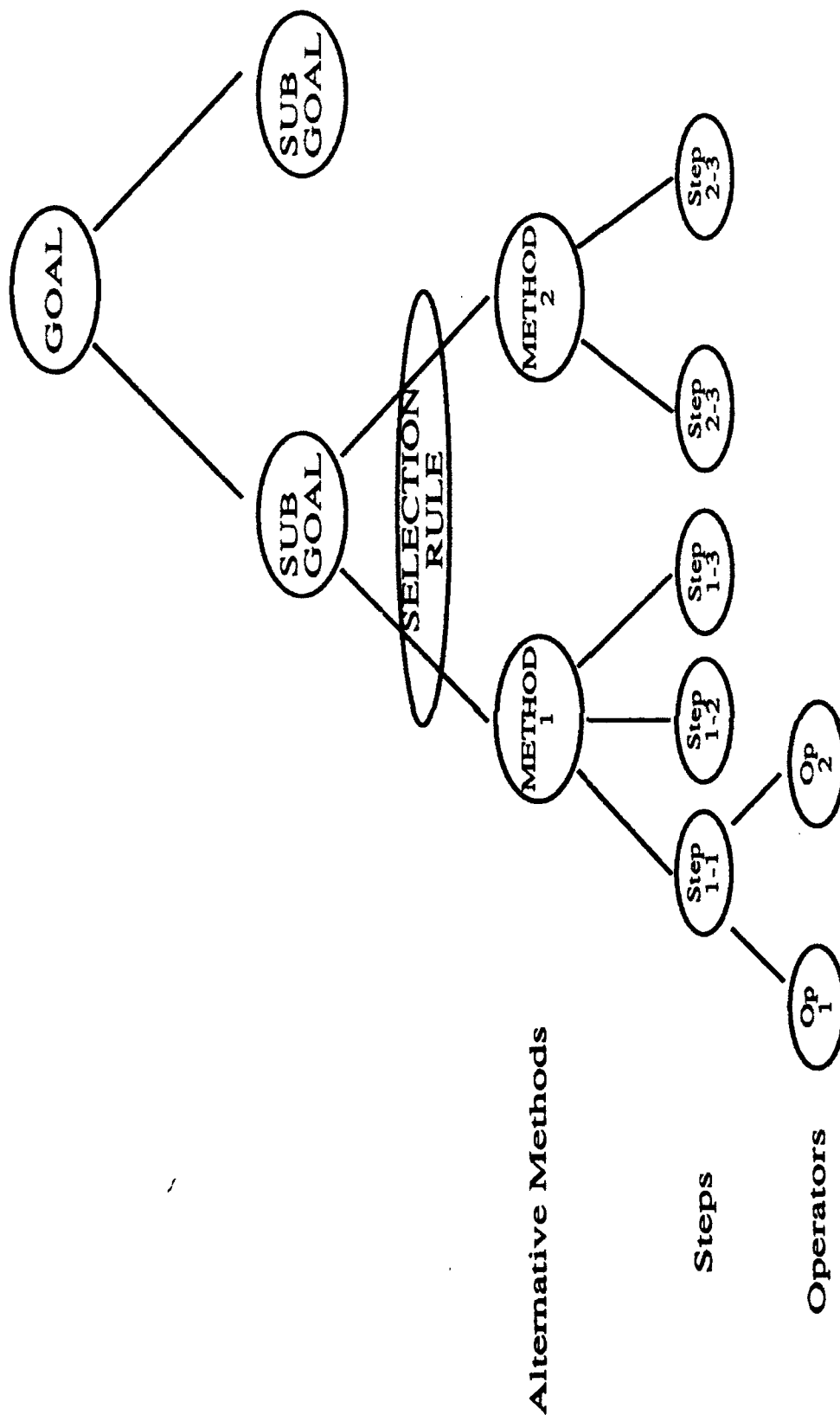


Figure 2 - An example of a GOMS hierarchy. (Abstracted from ideas of Card, Moran, and Newell, 1983; Kieras, 1988; Kieras and Polson, 1985; Williams, 1991)

have been defined, the process returns to the leftmost method in the tree. This is what is meant by breadth-first. The method steps are again transformed to subgoals for which new methods need to be described in order for the subgoals to be accomplished. This process continues until each step in a method is identified as a primitive operator (one which can no longer be decomposed) or a subgoal which the analyst does not wish to further decompose. When all steps in all methods are primitive operators, or subgoals which require no further decomposition, the analysis is complete.

The process for constructing a representation of "how-it-works" knowledge, as described by Miyake (1986), is similar if not identical to that for structuring "how-to-do-it" knowledge. In the Constructive Interaction case, a top-level function is specified as opposed to a top level goal. Instead of a top-level method made up of a set of high level steps or operators, a set of high level mechanisms is specified to enable the function. This set of mechanisms may have a sequence of actions, or some may be enabled in parallel. Each mechanism is then converted to a subfunction which must be realized by a set of submechanisms. It may also be the case that more than one set of submechanisms can be used to achieve the function. If this is the case, then a set of conditions or parameters must be specified for selecting one set versus another set of submechanisms. As a result of the similarity between these two techniques, simply changing the language used in the instructions of the interview process can yield qualitatively, "how-it-works" knowledge in the same fashion that the GOMS terminology is used to yield "how-to-do-it" knowledge. Our efforts to automate the development of models of procedural tasks can to a large extent be transferred to a process for automating the acquisition of qualitative cognitive models of physical systems. By changing the language used in eliciting knowledge from experts, the GOMS methodology can be translated into a method for eliciting knowledge units which model one's understanding of how a physical device works.

## **Automating the Cognitive Task Analysis Process**

Having provided a brief overview of this GOMS analysis process, a detailed description of the operationalization of this process in a software system called Cognitive Analysis Tool (CAT) is presented in the following. As this description unfolds, extensions to the original GOMS analysis technique are discussed. These extensions were provided to increase the flexibility of the GOMS technique by breaking the rigid hierarchy typical of GOMS. Consequently, the CAT system can model tasks which can be interrupted so that new subgoals can be set outside of the task hierarchy. This essentially handles exceptions to normal task execution. Additionally, a capability to handle the potential failure of steps to be executed was also provided to model recovery from the failure to execute a task, ensuring that careful consideration of all possible alternative methods has been given to the modeling process. Figures 3a-d depict a logic flow diagram of the CAT system.

CAT provides two modes of operation. One mode is referred to as the GUIDANCE mode and the other is a free play EDIT mode. The GUIDANCE mode constrains the user to follow a specified process for capturing knowledge about a cognitive task. The EDIT mode can be accessed at any time so that the user can control the sequence of activities he/she wishes to activate. Since it is assumed that most users will be unfamiliar with the cognitive task analysis process, the GUIDANCE mode of operation shall be discussed in detail in the order in which the user would encounter the various dialog boxes presented by the system. The EDIT mode allows access to all dialog boxes presented in the GUIDANCE model but does not impose a sequence on their presentation.

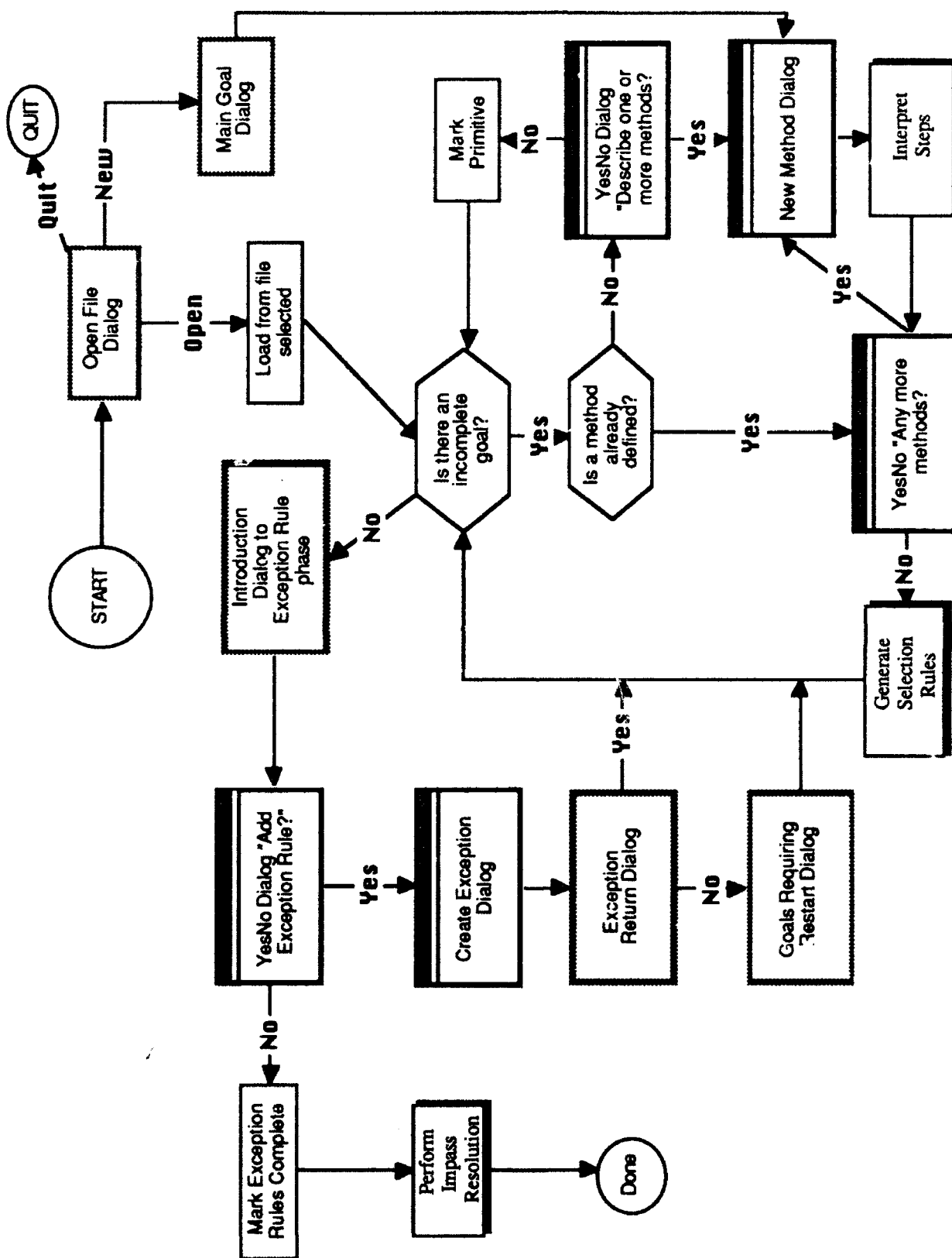


Figure 3a - Overall logic flow diagram for CAT

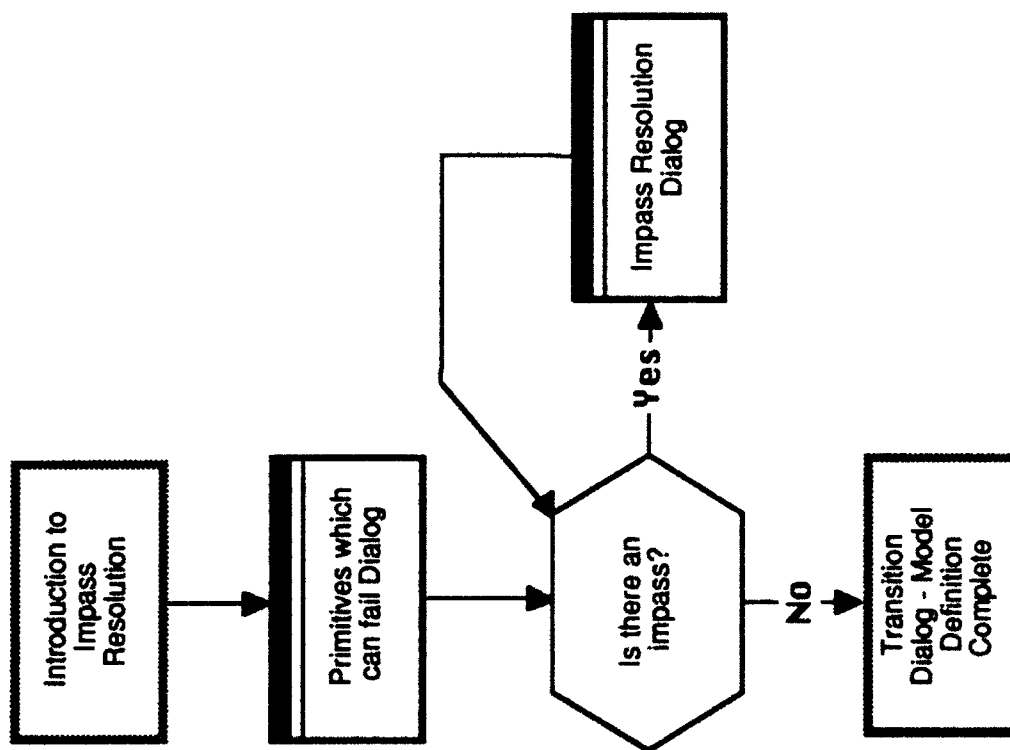


Figure 3b - Logic flow diagram for impass resolution subsystem in CAT

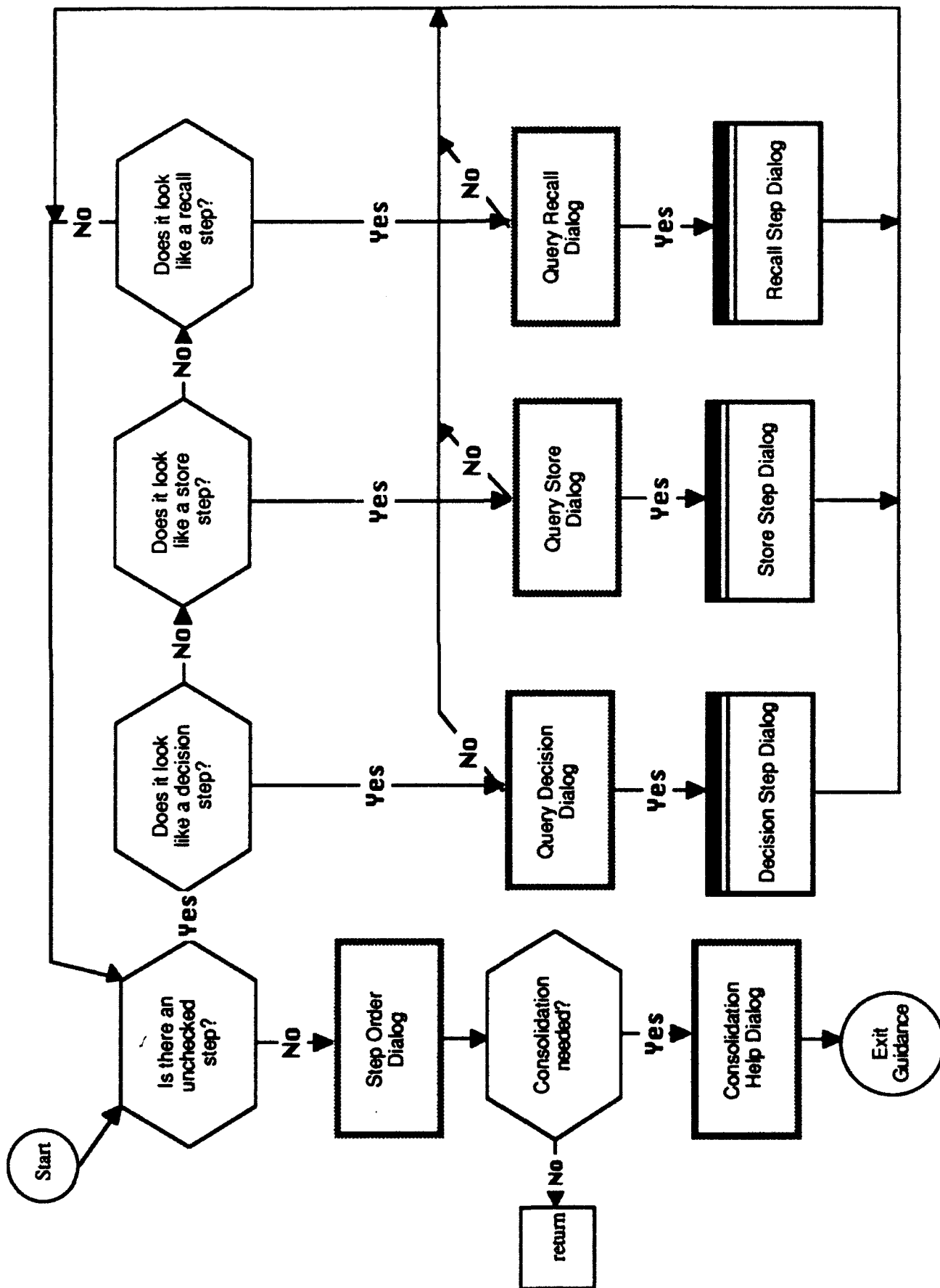


Figure 3c - Logic flow diagram for step interpretation subsystem in CAT

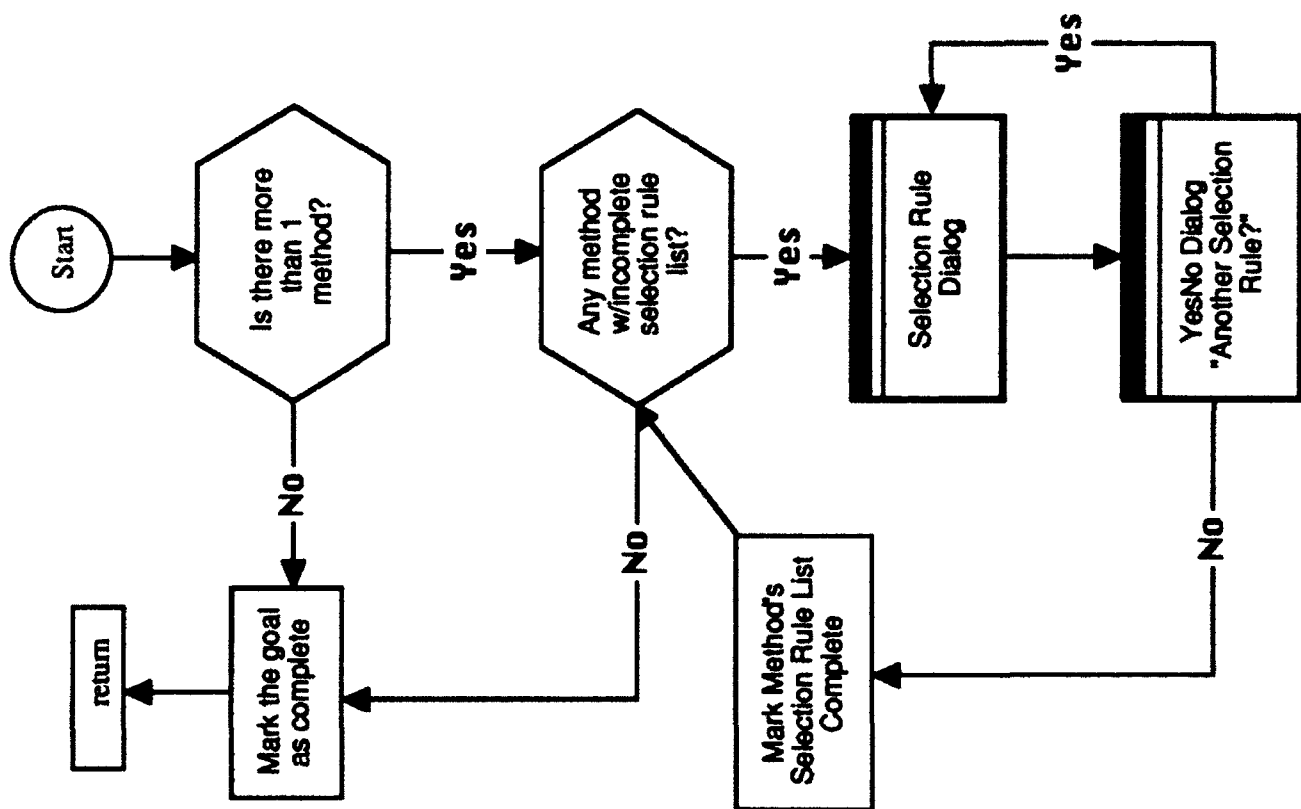


Figure 3d - Logic flow diagram for selection rule subsystem in CAT

## Opening a File

Upon selecting the CAT icon from the Windows 3.1 Program Manager, the first dialog box encountered is the Open File dialog box. We assume that the user is familiar with a graphical user interface and the operation of a mouse input device. Figure 4 depicts this dialog box. The user may select a model stored in a file or select NEW to create a new model. For purposes of describing the process it is assumed that NEW is selected. Having selected NEW, the Main Goal dialog box is presented to the user.

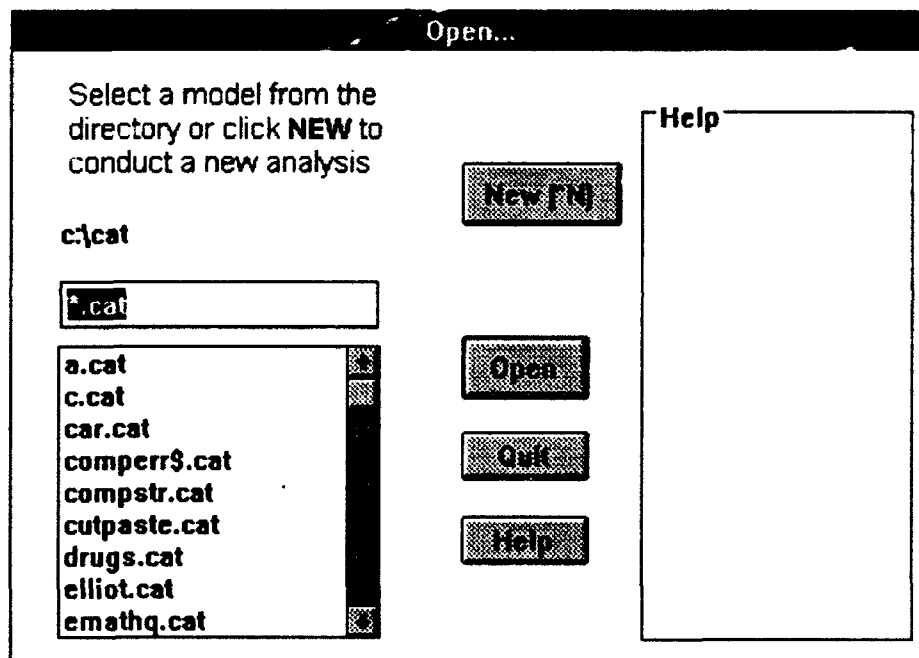
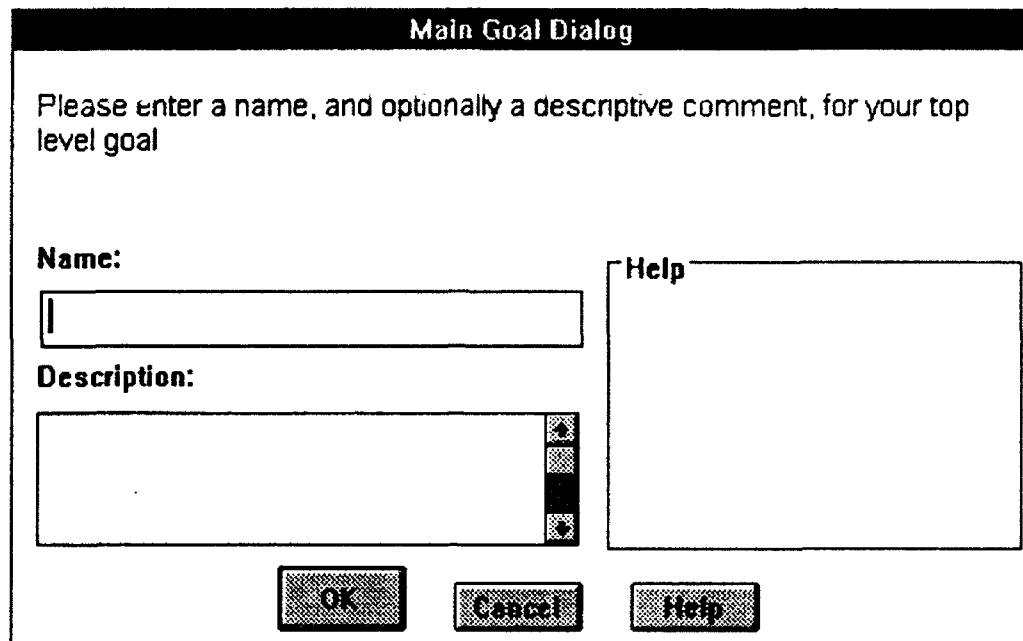


Figure 4 - Open File Dialog

## Specifying the Main Goal

As Figure 5 depicts, the user is requested to give a name for the top level goal to be achieved by the cognitive task model under development. A field for elaborating on the

description of the goal is also provided. Having specified a name for the top level goal, the user is instructed to click on OK.



The image shows a software dialog box titled "Main Goal Dialog". Inside the dialog, there is a text prompt: "Please enter a name, and optionally a descriptive comment, for your top level goal". Below this prompt, there are three input fields. The first is labeled "Name:" and is a single-line text box. The second is labeled "Description:" and is a multi-line text box with a vertical scrollbar on its right side. To the right of these two fields is a larger text area labeled "Help". At the bottom of the dialog box, there are three buttons: "OK", "Cancel", and "Help".

Figure 5 - Main Goal Dialog

### Creating a New Method

This action presents the user with the Create New Method dialog box as presented in Figure 6. The New Method dialog box requires the user to give the method a name and then to describe a set of steps which make up the method. The user may describe as many steps as he/she wishes. If there is only one method to accomplish the goal, the method name will typically be similar to the goal name.

**Create New Method...**

**Edit**

Enter the steps necessary to accomplish the goal 'do it'.

Method Name:

Step 1:

Help

New Insert Delete

OK Cancel Help

Figure 6 - New Method Dialog

### Decision, Store, Recall, and Go To Steps

Following this step description, each step is checked to determine if (1) the step described requires that a decision be made, (2) the step described requires that information needs to be stored in memory for later retrieval, or (3) the step requires that some information be recalled from a memory store.

If any step signifies that one of these three actions is implied the user is further queried about each of the appropriate steps. If a step description is indicative of a decision, the user is presented with the Query Decision dialog box of Figure 7. This dialog box determines if the user's step description implies that a decision must be made. If so, then the Decision Step dialog box of Figure 8 is presented. The Decision Step dialog box imposes one IF-THEN-ELSE structure upon the user which must be filled in to indicate the specifics of the decision to be made by the decision step specified.

Query Decision Dialog

One of the steps you entered was:

**if today is Sunday then go to church**

Your use of the word 'if' in this step indicates that a decision needs to be made at this point.

If the action to be performed at this point is dependent on some condition, press the button labeled 'Yes'. Otherwise, press the button labeled 'No'.

Help

Yes

No

Cancel

Help

Figure 7 - Query Decision Dialog

Decision Step Dialog

Please fill in the following information:

if

then

else

Help

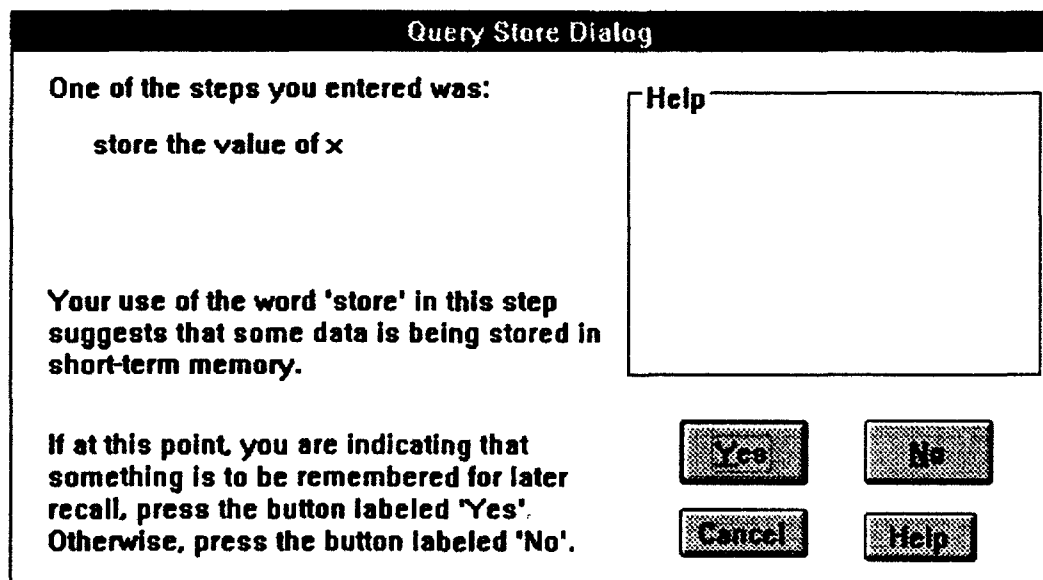
OK

Cancel

Help

Figure 8 - Decision Step Dialog

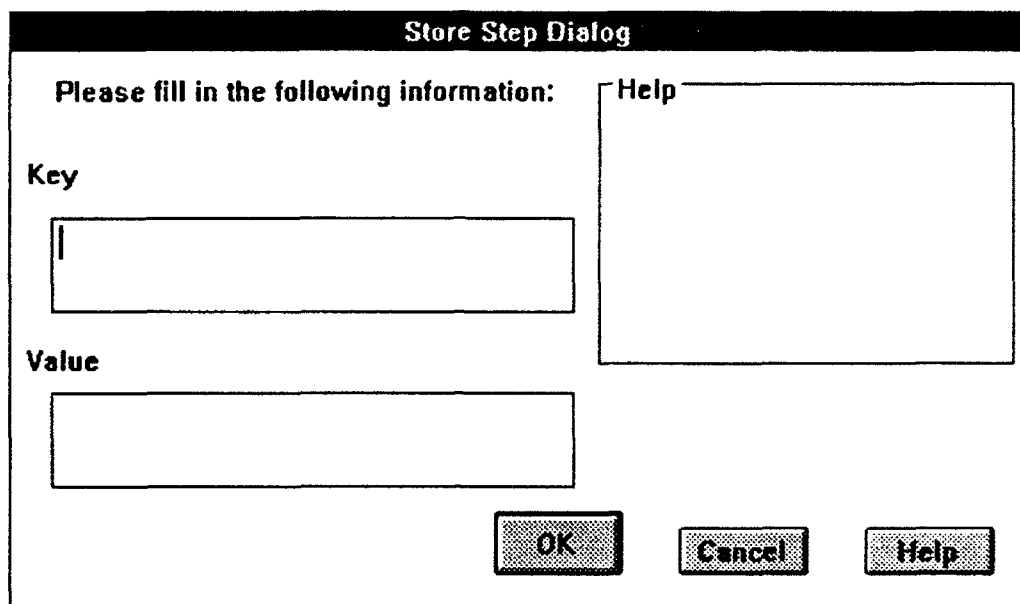
If a step description indicates that some information must be stored in memory, the user is presented with the Query Store dialog box as in Figure 9. The Query Store dialog box requests confirmation from the user concerning the correctness of the inference that a step implies that some information be stored in memory. If the implication is confirmed, the Store Step dialog box is presented to the user as depicted in Figure 10. The user is then requested to supply the appropriate information; namely a name or "key" under which the information is to be stored and the value to be stored under that name. For example, store the value "banana" under the key "fruit".



The image shows a software dialog box titled "Query Store Dialog". It contains the following text and controls:

- Title:** Query Store Dialog
- Text 1:** One of the steps you entered was:  
store the value of x
- Text 2:** Your use of the word 'store' in this step suggests that some data is being stored in short-term memory.
- Text 3:** If at this point, you are indicating that something is to be remembered for later recall, press the button labeled 'Yes'. Otherwise, press the button labeled 'No'.
- Buttons:** Yes, No, Cancel, Help
- Help Field:** A rectangular box labeled "Help" is located to the right of the first two text blocks.

Figure 9 - Query Store Dialog



The image shows a 'Store Step Dialog' window. At the top is a title bar with the text 'Store Step Dialog'. Below the title bar, the text 'Please fill in the following information:' is displayed. To the left of this text are two input fields: the top one is labeled 'Key' and the bottom one is labeled 'Value'. To the right of the 'Please fill in the following information:' text is a larger text area labeled 'Help'. At the bottom right of the dialog box are three buttons: 'OK', 'Cancel', and 'Help'.

Figure 10 - Store Step Dialog

If a step name implies that some information be recalled or retrieved from memory, the user is presented with the Query Recall dialog box of Figure 11. If the user responds that the step does imply a recall of some information from memory, then he/she is presented with the Recall Step dialog box of Figure 12. This dialog box like the Store Step dialog box requires that the user specify the name or "key" of the slot in memory from which information is to be recalled. These checks on step descriptions are conducted on each step of each method described during the analysis process. For example, if one wished to recall the value of "fruit", from the store-step defined, the system would return "banana".

Query Recall Dialog

One of the steps you entered was:

recall the value of x

Your use of the word 'recall' in this step suggests that some data is being recalled from short-term memory.

If at this point, you wish to indicate that some information, previously stored, is to be recalled, press the button labeled 'Yes', otherwise press the button labeled 'No'.

Help

Yes

No

Cancel

Help

Figure 11 - Query Recall Dialog

Recall Step Dialog

Please fill in the following information:

Key

Help

OK

Cancel

Help

Figure 12 - Recall Step Dialog

Additionally, the user may transfer control of step execution within a specific method simply by typing "go to" and indicating the step number within the method to which execution is to be transferred. "Go to" steps can be used as an individual step or in the context of a decision step of the IF-THEN-ELSE type. Having completed these checks and the appropriate responses required, the user is presented with the Step Order dialog box.

### **Ordering Steps**

The Step Order dialog box of Figure 13 requests the user to identify the order in which the steps of the method defined are to be executed. The choices are: (1) the order in which the user defined the steps, (2) any order or (3) some other order. If "As Entered" is selected, the system will assume the order in which the steps were defined in the New Method dialog box. If "No Particular Order" is selected, then the system will assume that the steps may be executed in any order. If "Other" is selected, the user is presented with the graphical portrayal of the method described. The user may then move the graphic boxes associated with any given step to the position in the graphic which indicates the order in which the steps are to be executed. The topmost step box of the method is the first step to be executed. That is, execution order is top down in the graphic representation of the method steps.

When interacting with the graphic portrayal of a method the user may also group steps by drawing a box around those steps to be grouped. Grouped steps are indicated by the gray shaded area surrounding a set of steps as depicted in Figure 14. The system then assumes that those steps which are grouped can be executed in any order or simultaneously. Simultaneous execution of the steps in a method is a capability which is

not assumed in a traditional GOMS model but has been introduced in CAT as a result of research conducted by Kieras and Meyer (1992) and John (1990). Simultaneous execution of steps is appropriate when modeling parallel tasks or a multi-tasking activity.

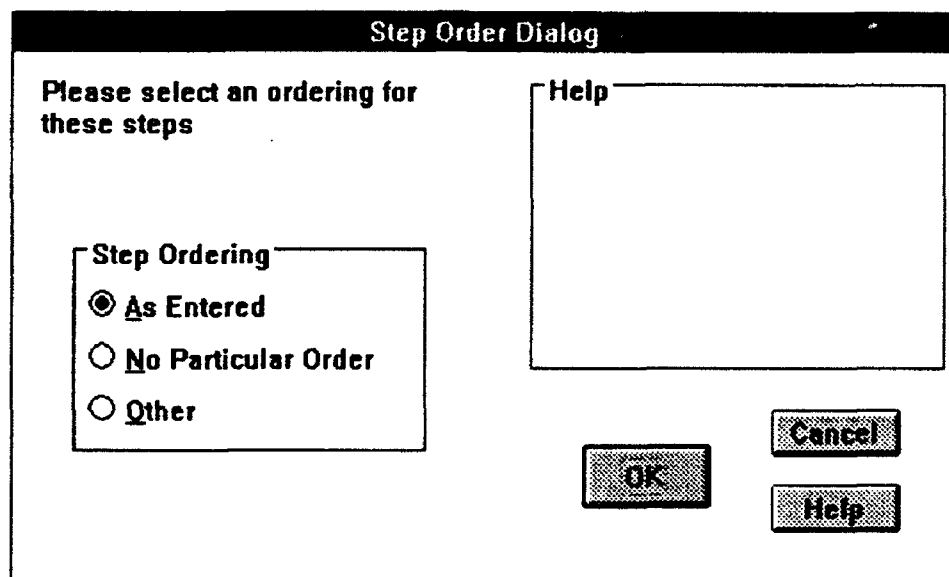


Figure 13 - Step Order Dialog

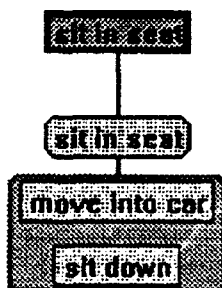


Figure 14 - Grouping Steps

Having specified an appropriate order for execution the system conducts a further check on the number of steps specified in a method. A decision was made early in the design process to allow the user to unpack or elicit as many steps in a method as he/she so

desired so as not to interfere with the recall process. However, concurrently, we also did not want the user to specify low level steps early in the modeling process. Consequently, it was decided that when the user has finished recalling steps in a method, if there were more than nine steps specified (the upper boundary on working memory capacity from Miller, 1956) the system would force the user to consolidate or chunk some of these steps into a higher level step. In order to do this consolidation, the user is presented with the Consolidation Help dialog box of Figure 15, which instructs the user as to the process for chunking some set of steps into a higher level step. In order to do this the user must exit the GUIDANCE mode, consolidate the step as instructed and then return to the GUIDANCE mode. Consolidate is accomplished by a method similar to that used when grouping steps for simultaneous execution. Consolidation is continued until not more than nine steps have been specified for any given method.

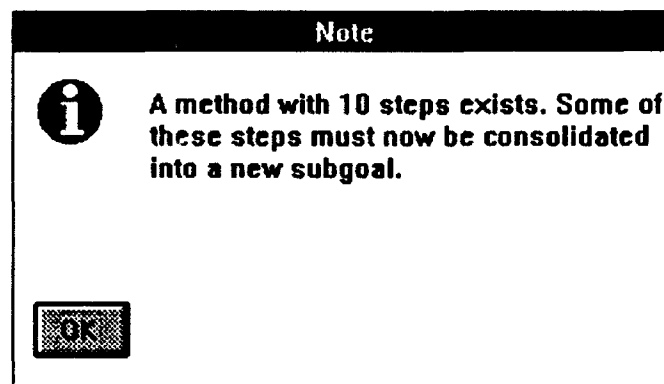


Figure 15 - Consolidation Help Dialog

### **Creating Alternative Methods**

After a method has been specified to achieve the top level goal, the system requests the user to specify any alternative methods associated with the accomplishment of

the goal as is depicted by the Any More Methods dialog box of Figure 16. If another method can be described, then the New Method dialog box of Figure 6 is presented again. The process iterates on the above mentioned sequence until the user responds "No" to the Any More Methods dialog box of Figure 16.

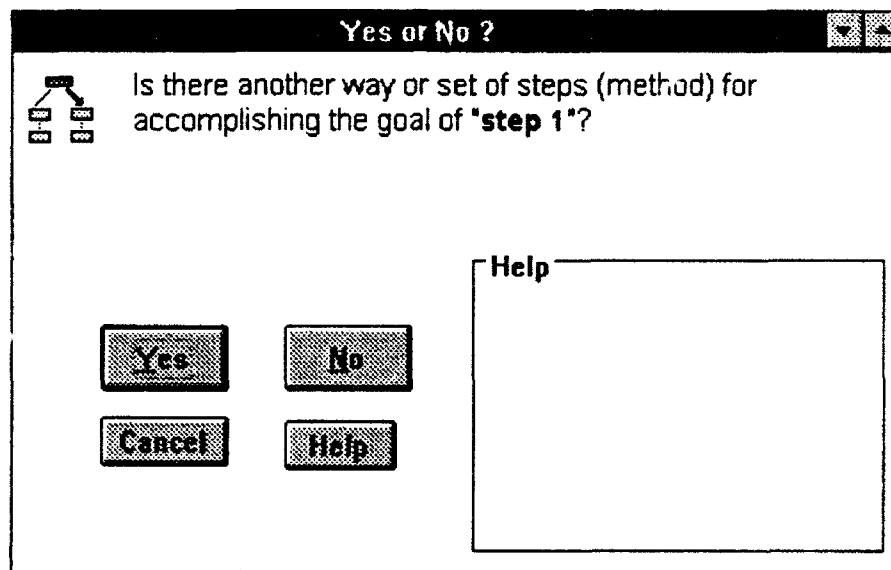


Figure 16 - Any more methods Dialog

### Creating Selection Rules

Having described a method or set of methods for a specified goal, the system will check to determine if more than one method was defined for the current goal. If there is not more than one method the goal is marked as completed. If there is more than one method which has been described, the system tests to determine if selection rules have been defined for each method. If any method is without a selection rule, the user is presented with the Selection Rule dialog box of Figure 17. This dialog box requests that the user define a condition or set of conditions which if all are true will trigger the

selection rule of the current method displayed in the display box. That is, conditions of the selection rule are 'and'-ed. However, there may be an alternative selection rule for the current method. Consequently, the user is presented with the Another Selection Rule dialog box of Figure 18. If another Selection rule can be described for a specific method the user will be returned to the Selection Rule dialog box of Figure 17 and requested to define another set of conditions. If more than one selection rule for a given method has been defined the selection rules are 'or'-ed. That is, depending upon the conditions of each alternative selection rule, one or the other may be triggered to select a specific method. The user may define as many selection rules as he/she wishes for any given alternative method in the set defined to accomplish a specific goal.

**Create New Selection Rule**

**Edit**

Please list conditions which, if all true, justify the use of the method 'use command key' for accomplishing the goal of 'issue the CUT command'

**Method Name:** use command key

**Condition 1:** user knows command key equivalents

**Help**  
title

New Insert Delete  
OK Cancel Help

Figure 17 - Selection Rule Dialog

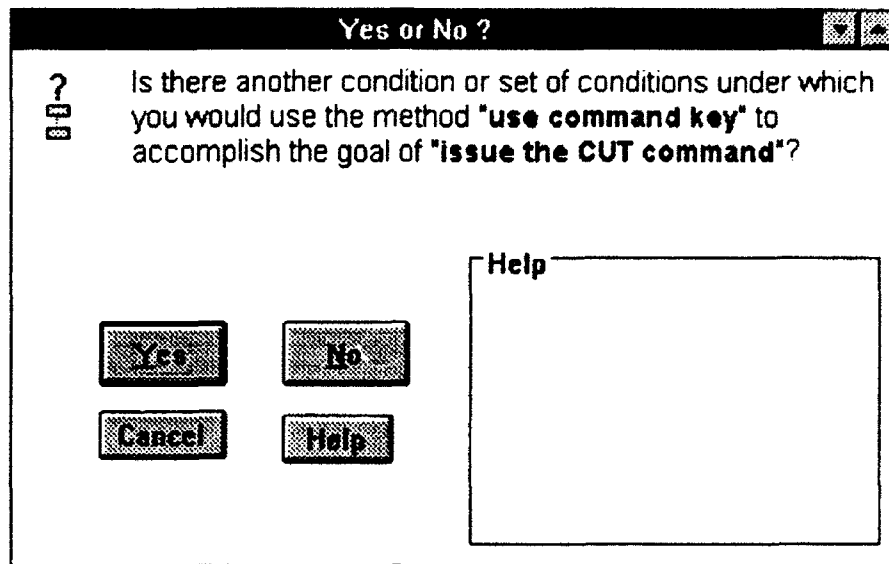


Figure 18 - Another Selection Rule Dialog

When selection rules have been defined for each alternative method under a specific goal, the system marks the selection rules for the methods "complete" and marks the specific goal "complete". The system treats each step of a method as a subgoal and determines if a method has been defined for that step. If not, the system presents the Describe Methods dialog box of Figure 19. If the user wishes to decompose the step as a subgoal along with a method or alternative methods, the New Method dialog box is presented as in Figure 6. If the user does not wish to decompose the step as a subgoal, the system marks the step as "primitive". Primitive steps are the lowest level steps which do not require further decomposition. The motivation of the system is to decompose all steps of all methods until all steps in all methods are marked "primitive". When all steps of all methods are marked primitive the first phase of the analysis is completed.

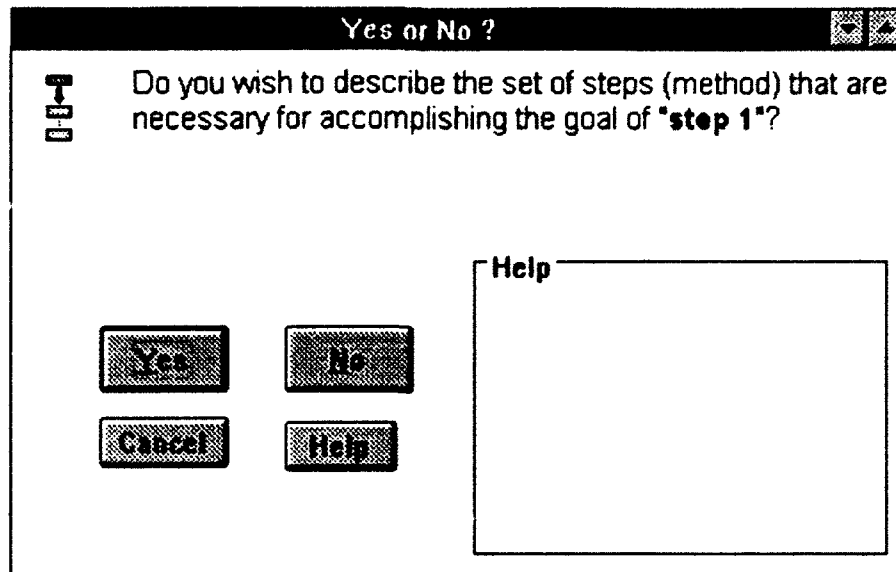


Figure 19 - Describe Methods Dialog

### Quitting the Process

At any time during the analysis process it is anticipated that the user may quit the modeling process. If so the name of the model as specified in the top level goal will be stored in a file for later retrieval. When the user returns to complete the model the Open File dialog box of Figure 4 is presented. The user may then select the appropriate file name, at which time, the model will be loaded into the system. The system will then check to determine if any steps need to be decomposed as subgoals. If so, the process described in the above is initiated.

## Creating Exception Rules

Given that all subgoals have been decomposed as a result of the first phase of the analysis, the user is presented with an Introduction To Exception Rules dialog box of Figure 20. This dialog box explains what is meant by an exception rule along with the purpose of exception rules. The user may request a more detailed explanation regarding exception rules by clicking on the HELP button. When the user has completed the introduction to exception rules, he/she is presented with the Add Exception Rule dialog box of Figure 21.

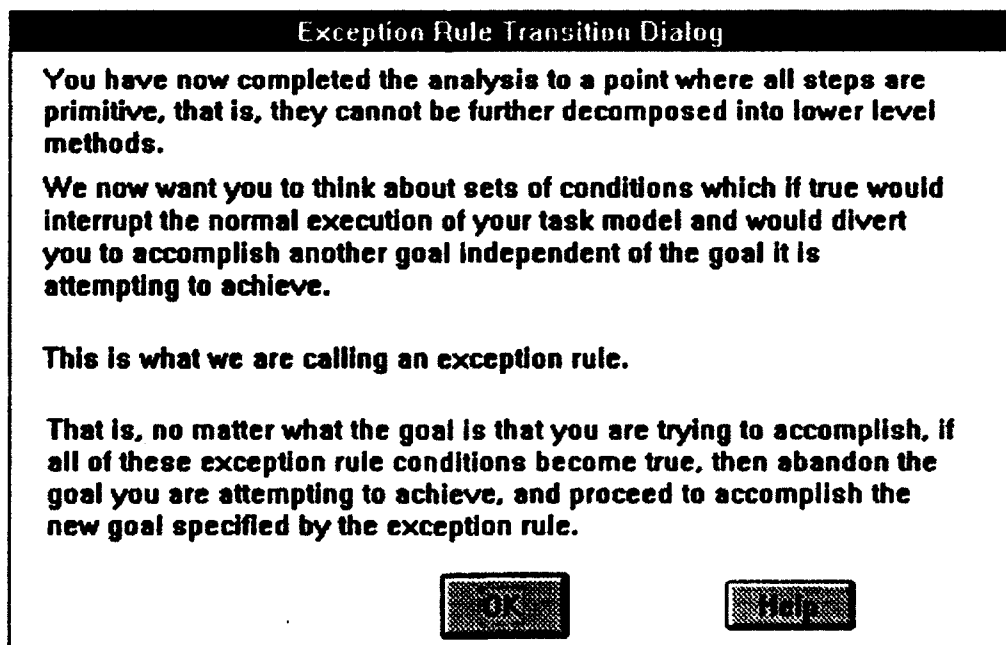


Figure 20 - Introduction to Exception Rules Dialog

The user is queried as to whether or not he/she wishes to define any exception rules for the model using the Add Exception Rule Dialog of Figure 21. If an exception rule is to be defined, the user is presented with the Create Exception dialog box of Figure 22.

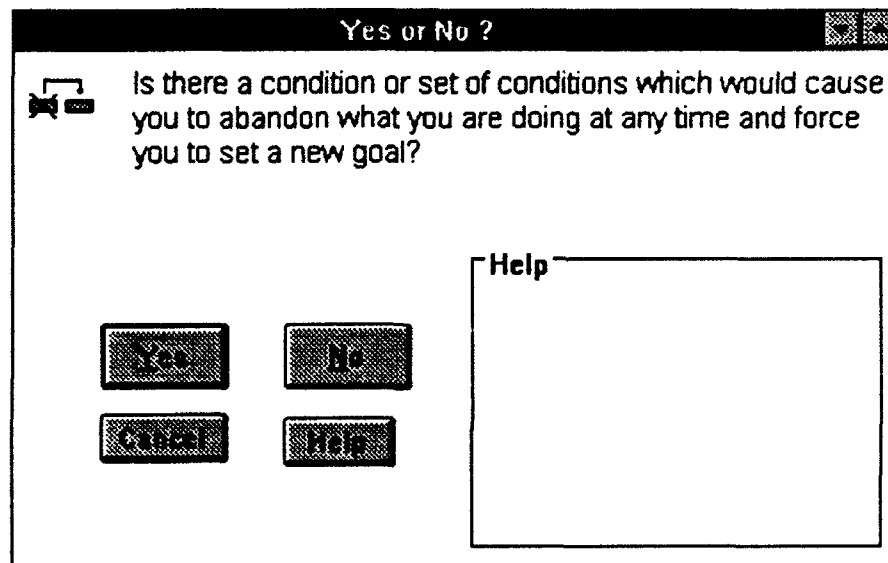


Figure 21 - Add Exception Rule Dialog

**Create New Exception Rule**

**Edit**

Please list conditions which, if all true, should trigger the execution of a new goal.

**Goal Name:**

**Condition 1:**

**Help**

**New** **Insert** **Delete**

**OK** **Cancel** **Help**

Figure 22 - Create Exception Dialog

The exception rule dialog box requires that the user indicate a set of conditions which if true would force the abandonment of task performance independent of what he/she is attempting to accomplish at the time the exception rule conditions were met. In brief, an exception rule is global and can be triggered at any time. It can be thought of as an interruption in normal task performance. Creating an exception rule also requires that the user specify a new goal to be set when exception conditions are met. This almost invariably will require that new methods, their steps and potentially new selection rules be developed to accomplish the goal which is set by the exception rule condition. The accomplishment of the exception rule goal may require the definition of a relatively complex model.

## Returning to Model After Completion of Exception Rule Goal

When the exception rule has been defined, the user is presented with the Exception Return dialog box of Figure 23.

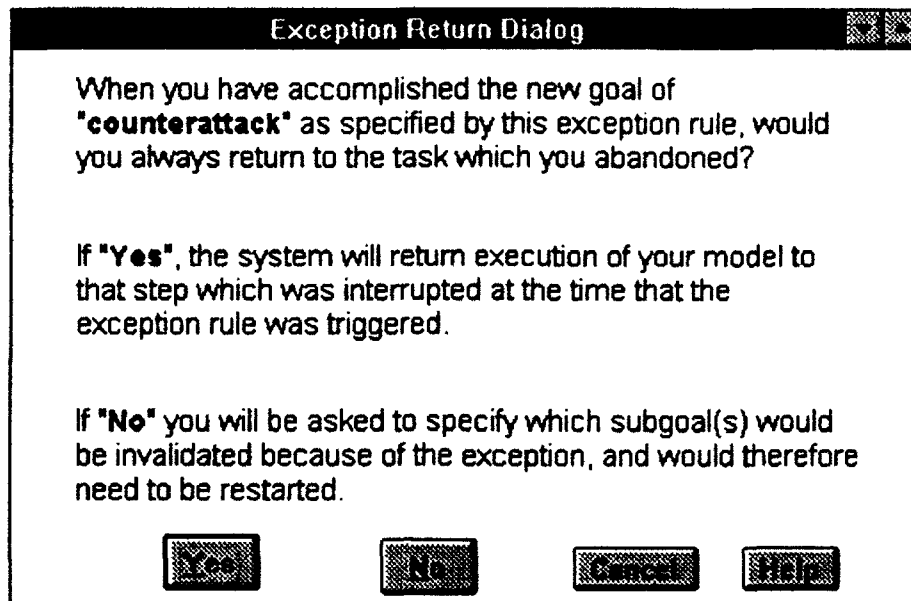
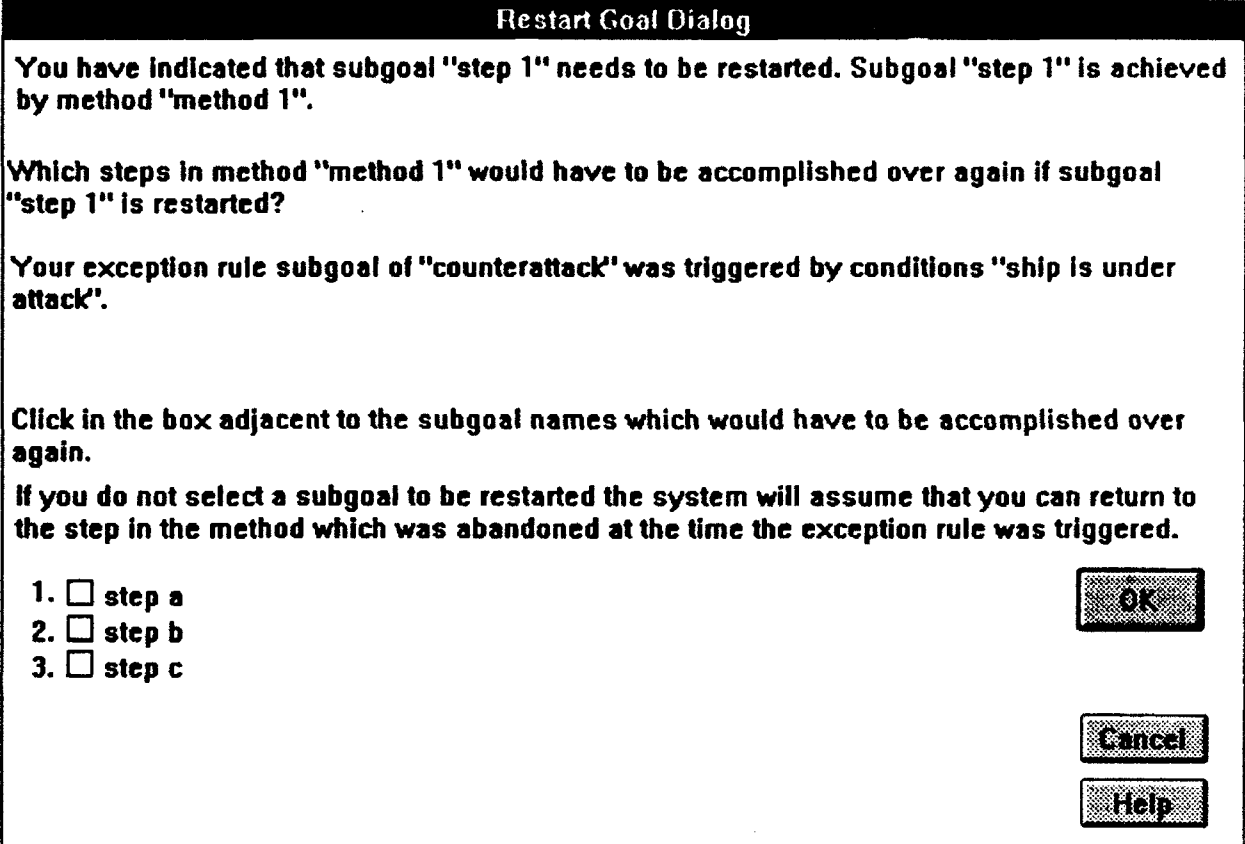


Figure 23 - Exception Return Dialog

As indicated by the dialog box the user is requested to think about where he/she would return in terms of normal task performance following the accomplishment of the exception rule goal. During execution of some task the user may simply return control back to the step which was abandoned when the exception was triggered. However, the world is dynamic and things change. Consequently, the user must determine if what was accomplished prior to the triggering of the exception rule has been undone with the passage of time. That is, the user must think about instances which could not be directly returned to following accomplishment of the goal specified by the exception rule. The best way to do this is for the user to think about subgoals which if abandoned would have

to be achieved over again if the task is interrupted. If this is the case the user is presented with the Restart Goal dialog box of Figure 24.



The dialog box has a title bar that reads "Restart Goal Dialog". The main text area contains the following information: a message stating that subgoal "step 1" needs to be restarted and was achieved by method "method 1"; a question asking which steps of method 1 would need to be repeated; and a note about an exception rule "counterattack" triggered by the condition "ship is under attack". Below this is a list of three subgoals: "step a", "step b", and "step c", each preceded by a checkbox. At the bottom right of the dialog are three buttons: "OK", "Cancel", and "Help".

**Restart Goal Dialog**

You have indicated that subgoal "step 1" needs to be restarted. Subgoal "step 1" is achieved by method "method 1".

Which steps in method "method 1" would have to be accomplished over again if subgoal "step 1" is restarted?

Your exception rule subgoal of "counterattack" was triggered by conditions "ship is under attack".

Click in the box adjacent to the subgoal names which would have to be accomplished over again.

If you do not select a subgoal to be restarted the system will assume that you can return to the step in the method which was abandoned at the time the exception rule was triggered.

1. ☐ step a

2. ☐ step b

3. ☐ step c

OK

Cancel

Help

Figure 24 - Restart Goal Dialog

This dialog box lists the subgoals of the model and requests the user to indicate which ones would have to be accomplished over again if they were interrupted. The list initially contains only the highest level subgoals of the model. The user is asked to click on the box next to a subgoal name which would have to be started over, if it were abandoned. For all those subgoals selected, the system will present the list of the next lowest level subgoals associated with those upper level subgoals selected. The user must then select which of these next lowest level subgoals might also need to be reached if its upper level subgoal was abandoned. The process continues to the next lowest level

subgoals associated with those selected at the immediately preceding upper level until the lowest level subgoals affected have been identified. The next list presented to the user consists of the second level of subgoals in the model which have not already been considered as a result of examining the first level of subgoals presented at the outset. This process continues as before until the impact of abandoning work on any subgoal as a result of an exception rule has been considered. This portion of the analysis is probably annoying to the user but is required to ensure that the user has considered all the possible effects of an exception rule being triggered. When the model building process is complete and a user wishes to execute a model this exception rule return information will tell the system where to go under any circumstance when the goal of the exception rule has been achieved. If no subgoals are specified for the return to the main model, the system automatically assumes that work on the task can be resumed at the step of the method which was interrupted because of the exception.

When the user has completed the process of defining where to return in the task model following accomplishment of the exception rule, the system again checks to determine if any goals/subgoals exist without specified methods. With the setting of a new subgoal as a result of creating an exception rule, the user is then required to describe a method or alternative methods to accomplish the new goal/subgoal. The process continues until all steps of all methods defined have been marked as primitives as was the case during the initial phase of the model building process. When all steps have been identified as primitives, the user is again queried to determine if any additional exception rules can be defined. If so then the exception rule creation process continues; if not the system marks all exception rules to be "complete".

## **Impass Resolution/Overcoming Failures**

The final phase of the analysis process is presented in the Introduction To Impass Resolution dialog box of Figure 25.

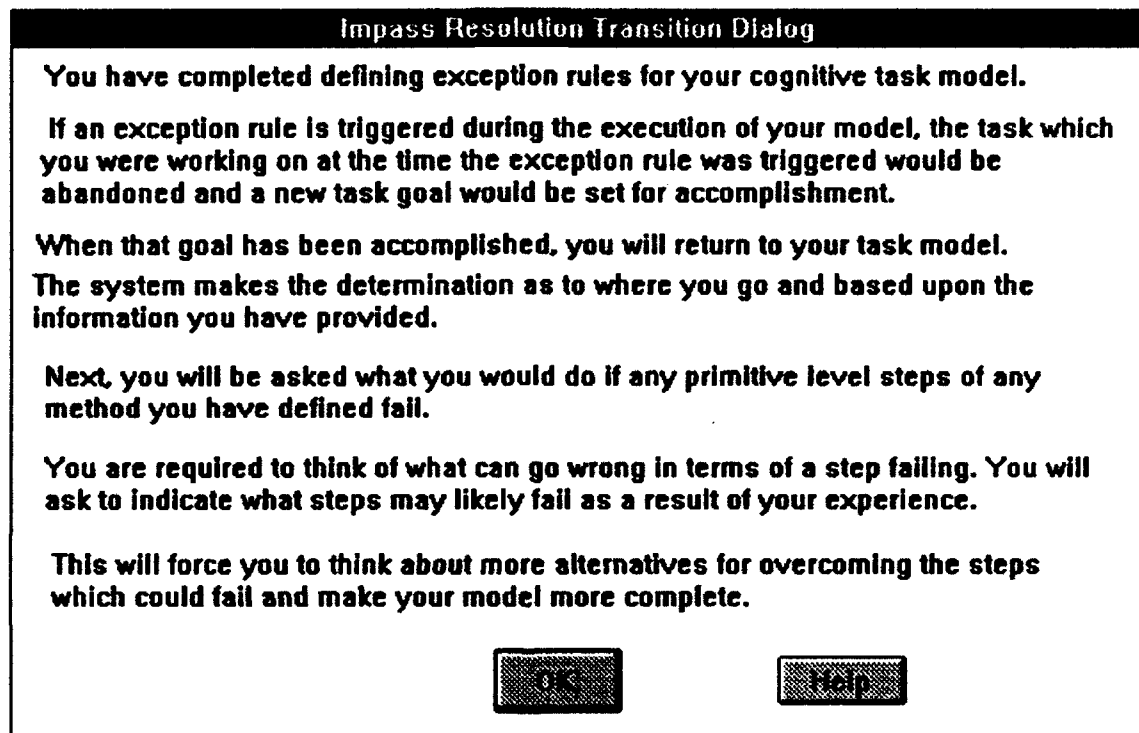


Figure 25 - Introduction to Impass Resolution Dialog

This phase is a further extension to the GOMS analysis process. An impass is a failed step. A step identified as an impass may potentially fail during the execution of the task model. Resolving such impasses requires that the user give careful consideration to assessing whether a sufficient and comprehensive set of alternative methods have been defined for each subgoal. If an appropriate alternative method has already been defined to cover the impass then the system will automatically select an alternative from those defined and attempt to accomplish the subgoal. This is referred to as back-tracking. If a method fails due to the failure of a step, the system will select the left-most method under

that subgoal if one exists and attempt execution. The system will continue in a left to right direction attempting to accomplish an alternative method which has been defined.

Following this Introduction To Impass Resolution, the system presents the Primitives Which Can Fail dialog box. This dialog box is presented in Figure 26.

**Primitives which can fail Dialog**

Please check all primitives which can fail

- ☐ determine position of beginning of text
- ☐ move cursor to beginning of text
- ☐ press mouse button down
- ☐ determine position of end of text
- ☐ move cursor to end of text
- ☐ verify correct text is selected
- ☐ release mouse button
- ☐ press the command key
- ☐ press the 'x' key
- ☐ release the 'x' key

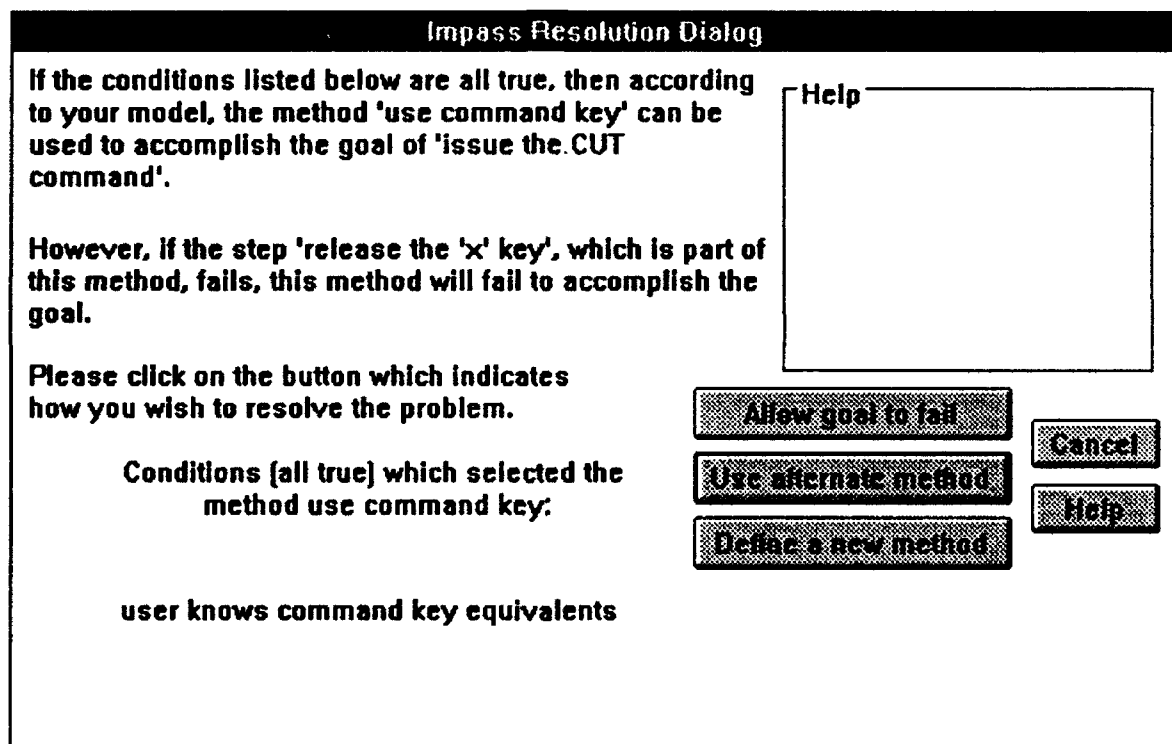
OK Cancel Help

Help

Figure 26 - Primitives which can fail Dialog

The Primitives Which Can Fail dialog box presents the user with a list of defined primitives generated during the model building process. The user is requested to click on the box(s) next to each defined primitive which can likely fail during the execution of the task model. If any primitive can fail the user is presented with the Impass Resolution dialog box of Figure 27. This dialog box presents the methods containing the primitive

step identified along with alternative methods for each subgoal to which the primitive step is related. The user may then examine the methods and subgoals which would be affected by the failed step. If a method is related to a subgoal which does not have any alternative methods, the user may allow that subgoal to fail or choose to develop another alternative method to overcome the impass if the step fails. If the user chooses to allow the subgoal to fail as a result of the impass and there are no alternative methods, the system will display the impact of the subgoal failure on the accomplishment of other higher level subgoals.



**Impass Resolution Dialog**

If the conditions listed below are all true, then according to your model, the method 'use command key' can be used to accomplish the goal of 'issue the CUT command'.

However, if the step 'release the 'x' key', which is part of this method, fails, this method will fail to accomplish the goal.

Please click on the button which indicates how you wish to resolve the problem.

Conditions [all true] which selected the method use command key:

user knows command key equivalents

Help

Allow goal to fail

Use alternate method

Define a new method

Cancel

Help

Figure 27 - Impass Resolution Dialog

If a step which can fail is part of a method whose subgoal has other alternative methods defined, these alternatives will be displayed. The user may then consider if an existing alternative method can be triggered to overcome the impass. If so, the user will

be requested to check the conditions of the selection rules for each potential alternative to determine if conditions of an alternative method are shared with that method which would fail as a result of the impass. If conditions do not match then the alternatives will not be selected by the back-tracking routine. The user may modify a selection rule for an alternative method via the edit selection rule command from the pull down menu if and only if such a modification is appropriate. If none of the predefined alternatives are appropriate, the user is instructed to select the create new alternative method choice or to allow the goal to fail. This process is repeated for each step which is selected by the user as a potential impass.

When all impasses have been addressed the user is presented with the Model Definition Complete dialog box as depicted in Figure 28 and instructed to respond appropriately.

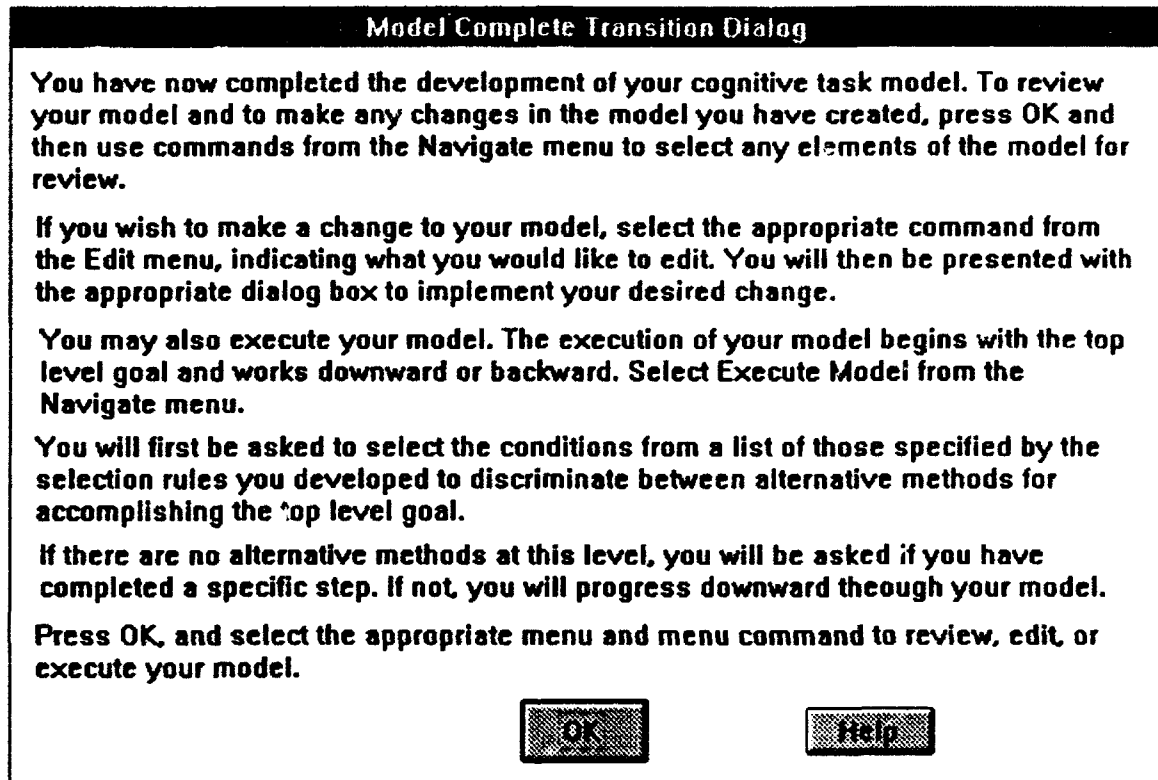


Figure 28 - Model Definition Complete Dialog

## Navigating and Editing a Model

When the user completes the development of a model, or when the user exits from the GUIDANCE mode of operation by canceling a dialog box, he/she will be able to view the various parts of the model in a graphical representation. The following presents a list of ways in which the user may view the various parts of the model.

### Viewing the Model

To view the top level goal, the user must pull down the "Navigate" menu as depicted in Figure 29 and select the item entitled "View top level Goal." This action will display the top level goal of your model and any method or methods associated with it. Alternatively, selecting "View Exception Rules" from the navigate menu will display the exception rules of the model.

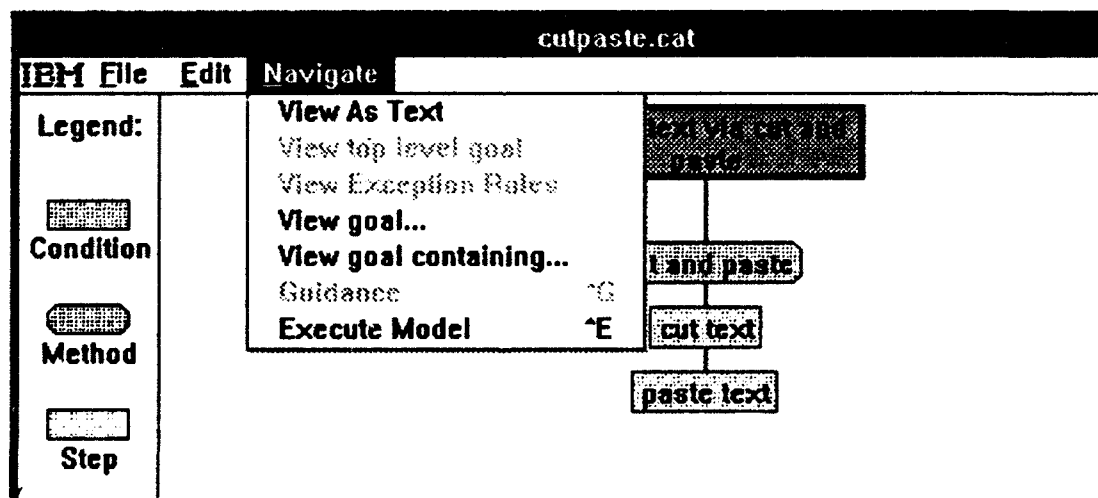


Figure 29 - The navigate menu

When reviewing any graphical representation of a specific model, the user may double click on any node representing a subgoal for which a method or methods have been defined. This action will result in the display of the methods defined for accomplishing that subgoal. This will move the user deeper into the graphical representation of the model.

At any time when reviewing a model, except when reviewing the top level goal of a model, the user may double click on the node at the top of the screen which contains the name of a subgoal. This action allows the user to view a rule which uses that subgoal as a step in a higher level method(s).

By selecting "View Goal ..." from the Navigate menu, the user will be presented with a list of all the subgoals defined in a specific model in the View Goal dialog box as in Figure 30. Choosing one of these subgoals, and then clicking on the OK button results in the selected subgoal's rules being displayed.

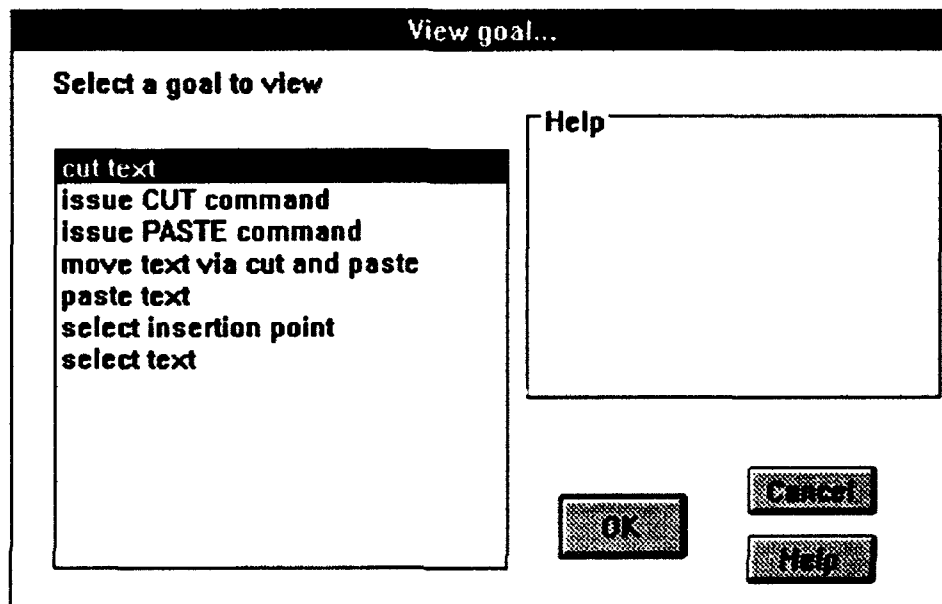


Figure 30 - View Goal Dialog

By selecting the "View Goal Containing ..." from the Navigate menu, the user will be presented with a list of all the steps in the model in the View Goal Containing dialog box of Figure 31. Choosing one of these steps and clicking on the OK button causes the system to present a list of all of the goals/subgoals which contain that step in the View Goal dialog box of Figure 30. The flow diagram for these View Goal dialog boxes is presented in Figure 32a.

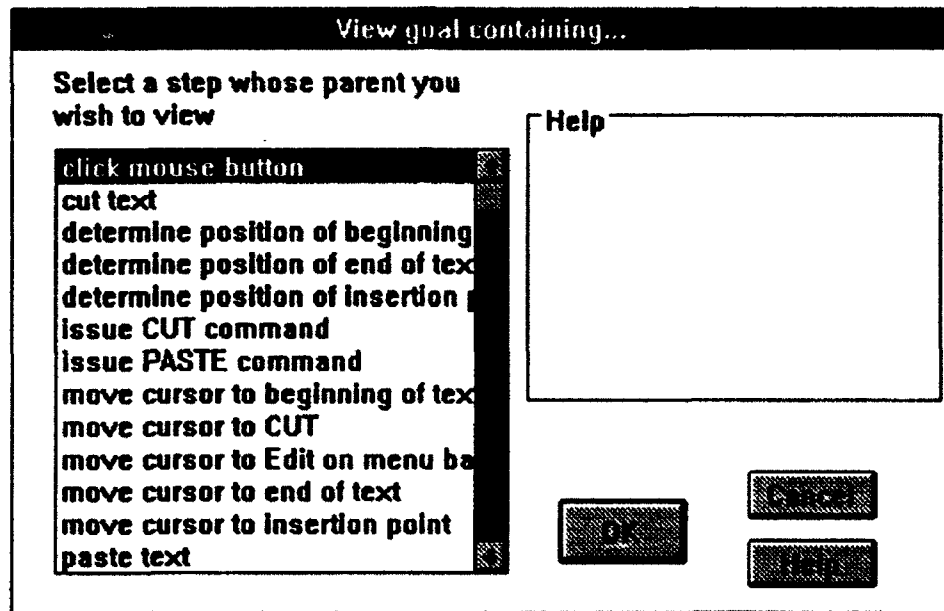


Figure 31 - View Goal Containing Dialog

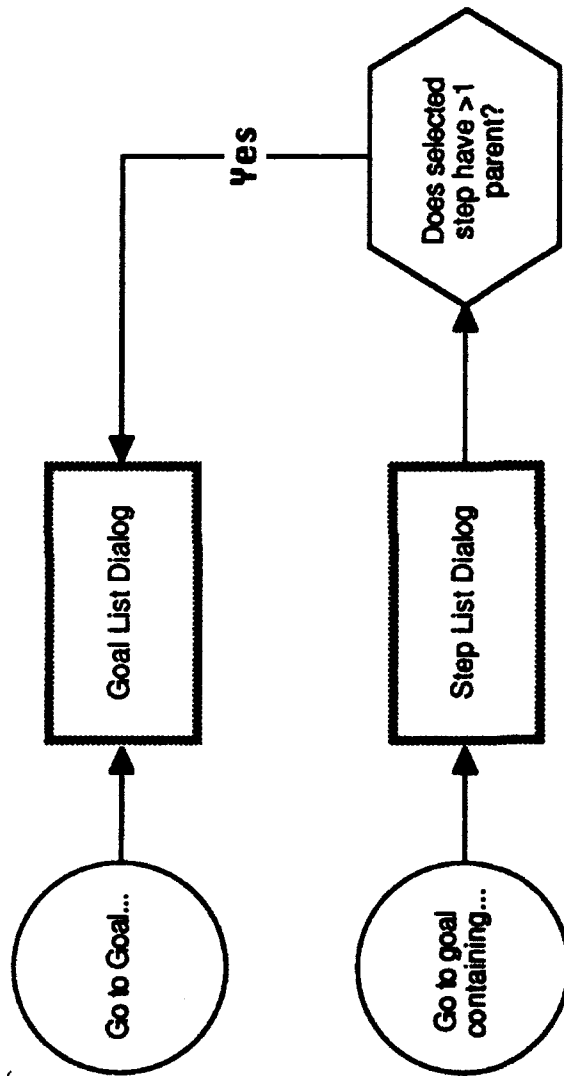


Figure 32a - Logic flow diagram for View Goal Menu Commands in CAT

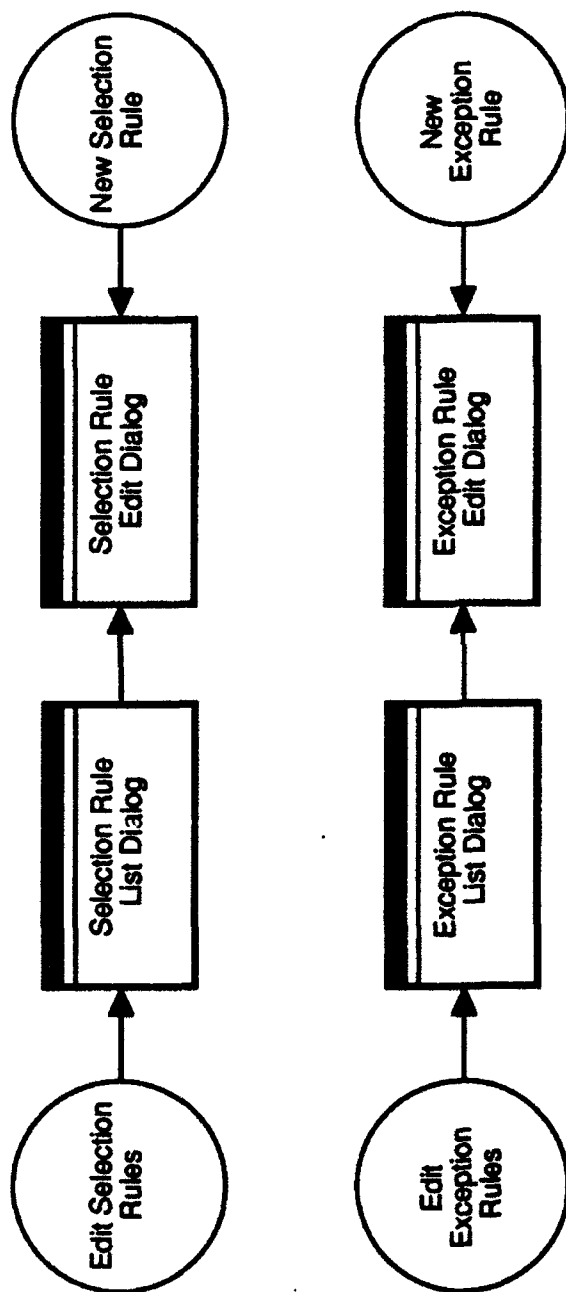


Figure 32b - Logic flow diagram for Selection Rule and Exception Rule Menu Commands in CAT

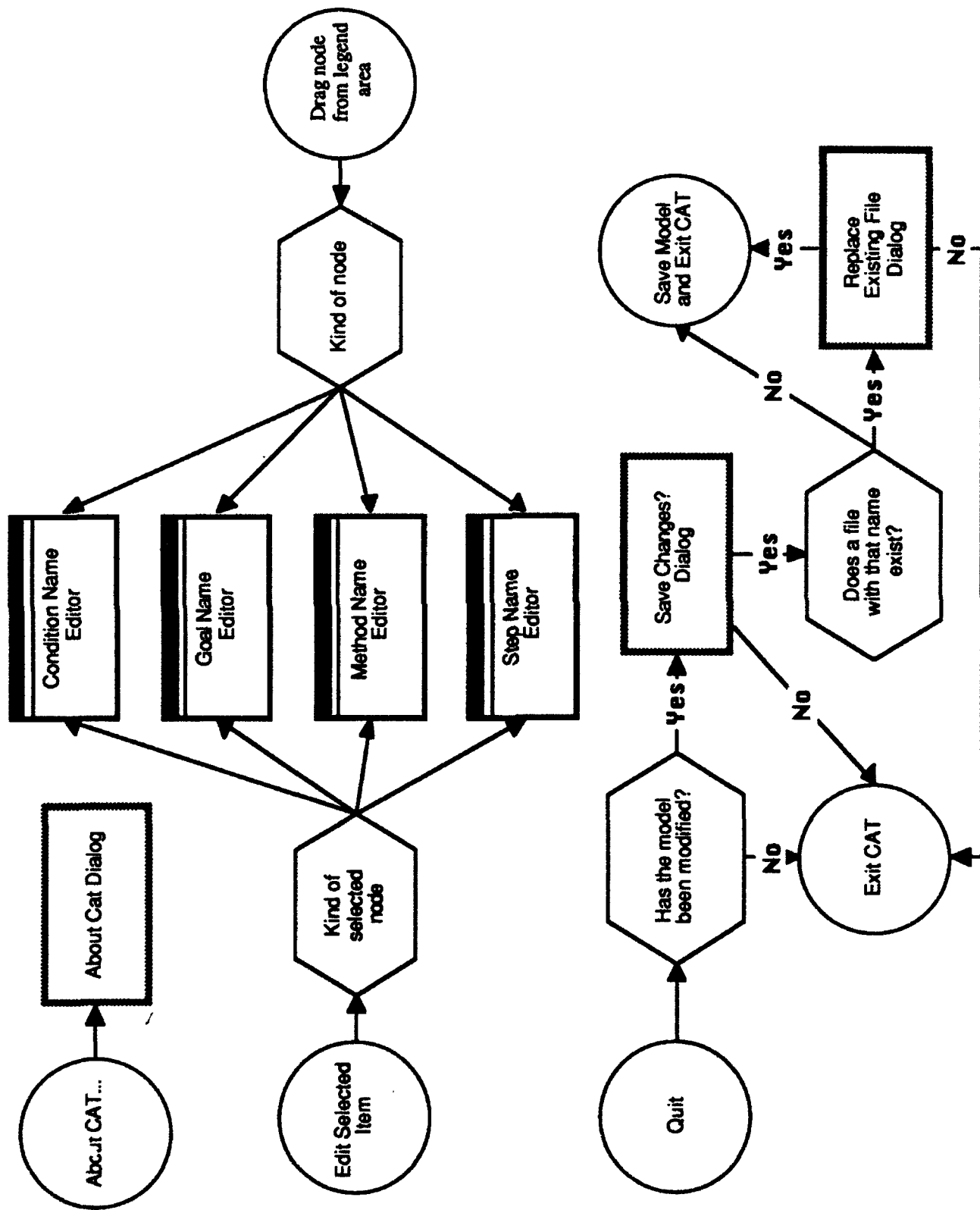


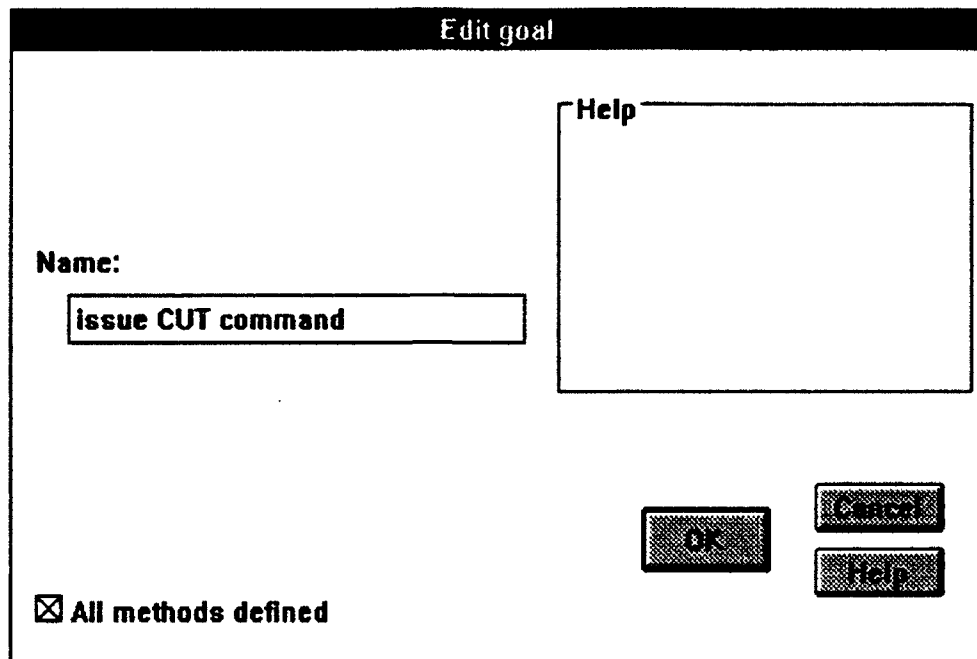
Figure 32c - Logic flow diagram for Editing and Quitting Menu Commands in CAT

## **Making Changes to a Model or Editing**

The logic flow diagrams for editing a model in the CAT modeling system are presented in Figure 32b, and 32c. The user can change the name of any goal, condition, method or step in a model by following these simple steps:

1. Display the rule containing the item whose name is to be changed using the Navigate capabilities of CAT.
2. Select the item on the graph by clicking on it.
3. Move to the Edit menu, pull down the menu and choose Edit Selected Item.
4. Make the desired changes in the dialog box presented.
5. Click on the OK button of the dialog box when finished making changes.

The dialog boxes presented to the user for making edits to a goal name, condition name, method name or step name are presented in Figures 33, 34, 35 and 36 respectively. In addition to the name changes which can be made while interacting with these dialog boxes, the user will be presented with an area in which notes can be entered. The notes placed in these dialog boxes become attached to the goal, condition, method or step being edited. Considerable text can be entered in the note areas of these dialog boxes. When the user is reviewing a model in its graphical representation these notes will be displayed in the lower right box of the displayed portion of a graph when the cursor is placed on any node of the graph.



The 'Edit goal' dialog box features a title bar at the top. On the left, there is a 'Name:' label above a text input field containing 'issue CUT command'. Below this is a checked checkbox labeled 'All methods defined'. On the right side, there is a 'Help' label above a large, empty rectangular area. At the bottom right, there are three buttons: 'OK', 'Cancel', and 'Help'.

**Edit goal**

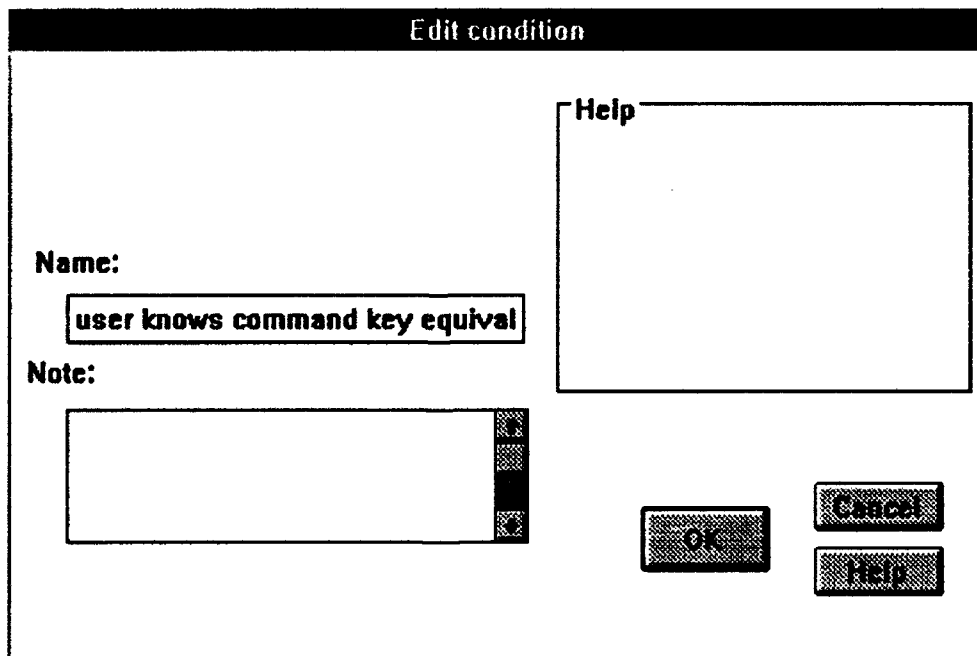
Name:  
issue CUT command

☒ All methods defined

Help

OK Cancel Help

Figure 33 - Goal Edit Dialog



The 'Edit condition' dialog box has a title bar. It includes a 'Name:' label with a text input field containing 'user knows command key equival'. Below this is a 'Note:' label with a text input field. On the right, there is a 'Help' label above a large, empty rectangular area. At the bottom right, there are three buttons: 'OK', 'Cancel', and 'Help'.

**Edit condition**

Name:  
user knows command key equival

Note:

Help

OK Cancel Help

Figure 34 - Condition Edit Dialog

**Edit method**

**Name:**

**Note:**

**Help**

Figure 35 - Method Edit Dialog

**Edit step**

**Name:**

**Note:**

**Help**

☒ **Primitive** ☐ **Can Fall**

Figure 36 - Step Edit Dialog

The user can edit (i.e. modify, delete, insert, etc.) any selection rule for a goal or subgoal being displayed by choosing "Edit Selection Rules ..." from the "Edit" menu. The Selection Rule List dialog box of Figure 37 is presented which allows the user to view all of the selection rules for the current goal or subgoal at the top of the graph displayed. This dialog box contains buttons for editing, deleting, or inserting a new selection rule. If a specific goal or subgoal has only a simple method then there is no selection rule to be displayed and the menu item will not be selectable. By clicking on the arrows above the box presenting the selection rules, the user can move through the list of selection rules related to the goal or subgoal which appears as the top goal/subgoal of the portion of the graph displayed.

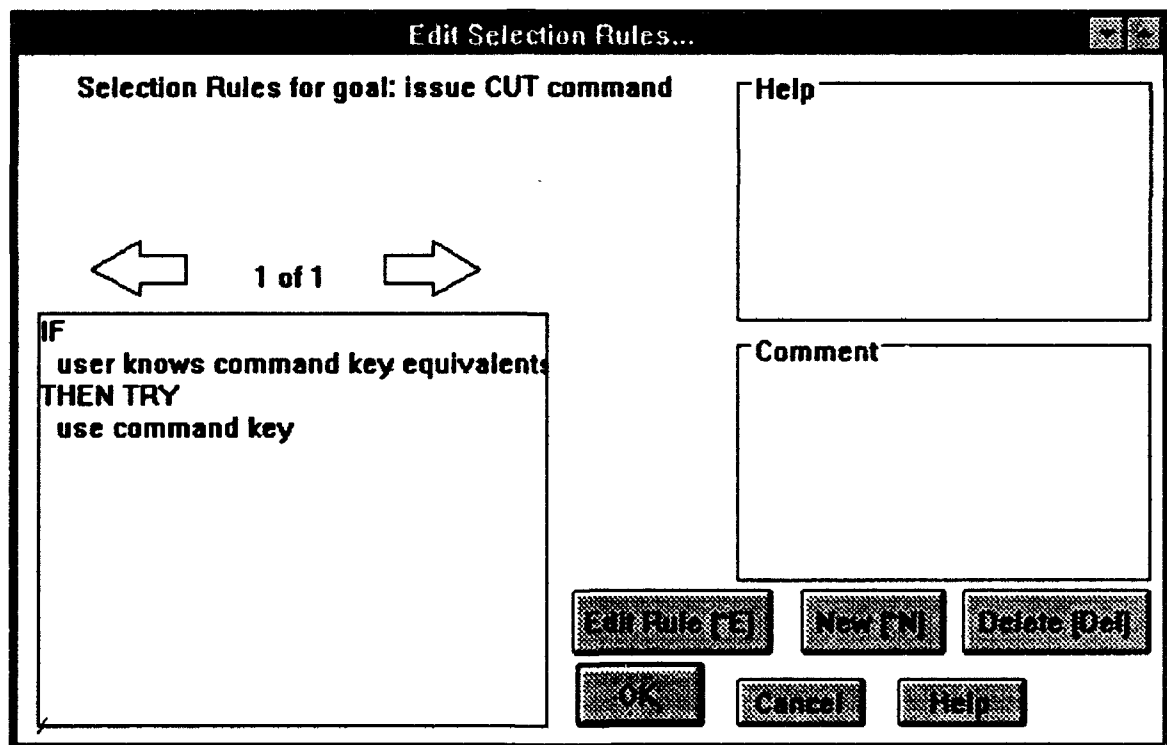


Figure 37 - Selection Rule List Dialog

The same process described above for editing selection rules applies to the editing of exception rules. To edit (i.e. modify, delete, insert, etc.) any exception rules for a specific model, the user can choose "Edit Exception Rules ..." from the "Edit" menu. Upon choosing this menu item, the user is presented with the "Exception Rule List" dialog box of Figure 38. This dialog box presents a list of all exceptions and contains buttons for editing, deleting or inserting a new rule.

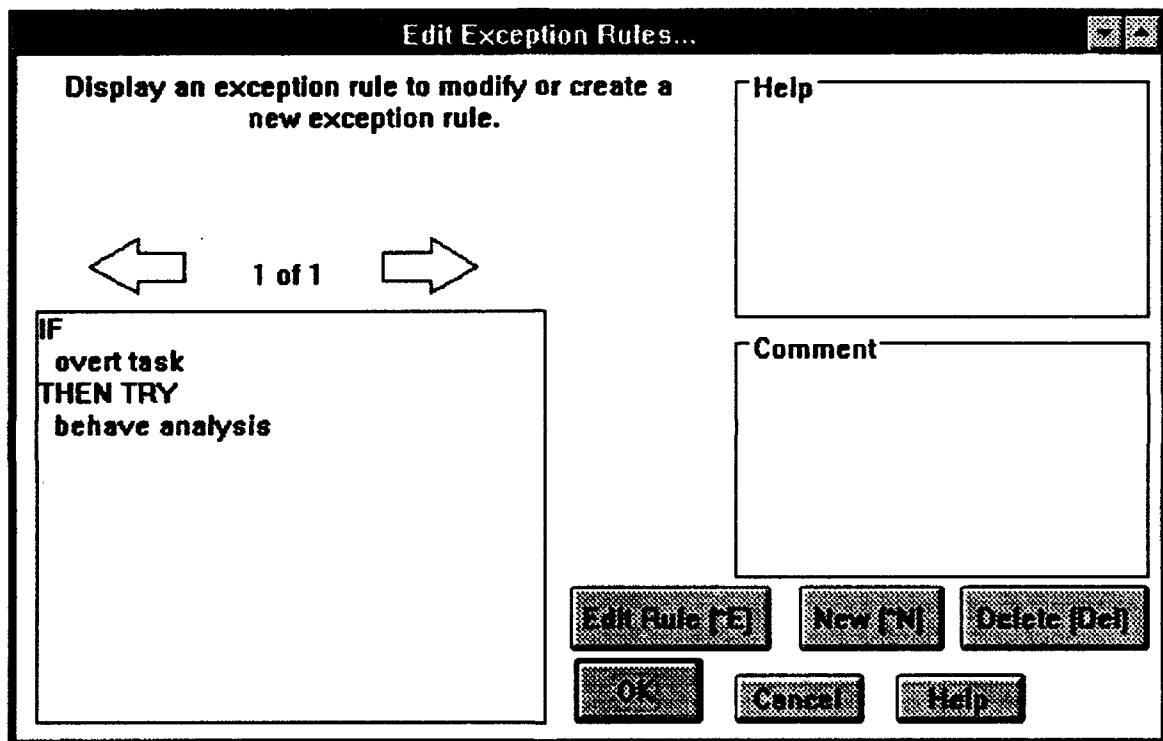


Figure 38 - Exception Rule List Dialog

### Changing Step Order

To modify the ordering of steps of any method, the user must display the graphic portrayal of the method he/she wishes to edit. The order of the steps of the method

displayed can be changed directly by clicking the mouse on a step whose position in the graph needs to be changed, and while continuing to hold down the mouse button drag the step vertically until the step is in the desired location in the method.

To indicate that a consecutive series of steps can be executed in any order or in parallel if possible, the steps can be grouped employing the graphical user interface of the system. To group a series of steps, the following must be performed:

- 1) Select the steps using one of the following two methods:
  - a) Enclose the steps in a rectangle by clicking the mouse down, in a position outside, above, and to the left of the first step to be grouped. Then drag the mouse with the mouse button down to a position outside, below and to the right of the last step to be grouped.
  - b) Select the first step by clicking on it. Then select subsequent steps by clicking on them while holding down the shift key.
- 2) After selecting the steps by either method described above choose "Group Simultaneous Steps" from the "Edit" menu. This will complete the grouping process.

If the user wishes to ungroup a series of steps which have been grouped, the user must select the steps to be ungrouped using one of the methods described above and choose "Ungroup Simultaneous Steps" from the "Edit" menu.

#### **Adding a New Method to a Model**

The user may add a new method to the goal/subgoal currently being displayed by choosing "Create New Methods ..." from the "Edit Method" menu. The user is then

presented with the "New Method" dialog box which was previously described for the GUIDANCE mode of operation. This dialog box allows for the creation of a new method and its accompanying series of steps.

### **Adding a Step to a Method**

A new step can be added to an existing method for the goal/subgoal currently being displayed by selecting any method node or step node on the display, indicating the location after which the new step is to be inserted, and then choosing "Create New Step ..." from the "Edit" menu. The user is presented with the Step Name Editor dialog box which was discussed in the above under the "Edit Selected Item" command of the "Edit" menu. The new step will be added directly under the method node selected or the step node selected.

### **Consolidating Steps into a New Method**

The user may also wish to consolidate some of the steps making up a method. During model development if more than nine (9) steps are defined for any given method, he/she is forced to go through a step consolidation process. However, during the editing phase of model development, the user may wish to consolidate some of the steps in an existing method. To do this, the user must select the steps he/she wishes to consolidate using one of the two methods described for grouping steps. Then the user must choose "Consolidate Steps" from the "Edit" menu. The steps will be consolidated into a new method and the user will be prompted to enter a name for the goal/subgoal of the new method, as well as a name for the new method.

## Graphical Editing

A much more direct process for editing a condition, goal, method or step has been incorporated into the CAT modeling system. This process is a graphic editing process. For any given portion of a graph currently displayed, the user can insert or delete a node. To insert a node the user must select the appropriate type of node to be inserted from the legend on the left column of the display depicted in Figure 39. While holding down the mouse button, the node can be dragged from the legend area and inserted in the appropriate position on the graph displayed. To delete a node from the graph the user can select the node by clicking on it with the mouse and then pressing the delete key on his/her keyboard. Alternatively, the user can select the node on the graph and then choose "Copy" or "Cut" from the "Edit" menu. "Cut" will delete the node from the display and place the contents of the node in the clipboard. Copy places the contents of the node in the clipboard but does not delete it from the display. The node placed in the clipboard can be placed on the display by selecting "Paste" from the "Edit" menu which will result in converting the cursor into the node such that the node can be placed in the appropriate location of the graph; clicking the mouse button completes the paste process. It is important to note that the top level node on the graph currently being displayed cannot be edited using this graphical interface capability.

Having moved a node to the appropriate location on the graph displayed, upon releasing the mouse key a dialog box is displayed. The dialog box displayed represents the type of node inserted and requests that the user enter the appropriate information relative to the new node inserted. The dialog boxes presented using this graphical editing method

are identical to those which are presented to the user as a result of editing activities enabled by choosing the "Edit Selected Items ..." command from the "Edit" menu.

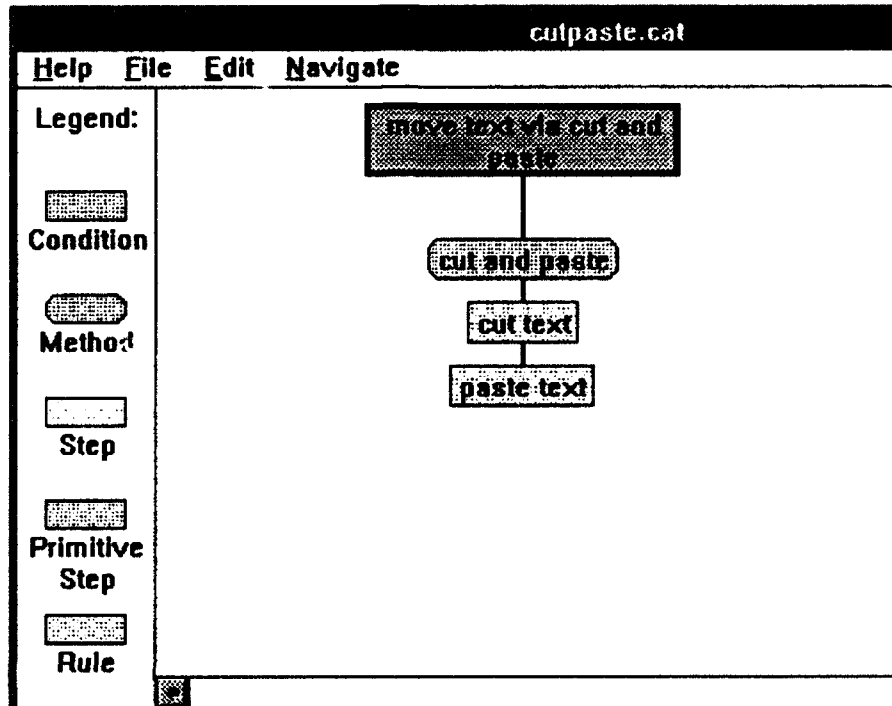


Figure 39 - Graphic Editing

### Executing A Model

The execution mode of the CAT modeling system allows the user to run the program created as a result of the modeling process in order to verify the accuracy of his/her analysis. Executing a model is a facility which can be used to guide someone through the procedures required to perform a task. This mode of operation also allows the user to enter rather detailed descriptions of each primitive level step which must be executed to accomplish the goal of the task modeled. The execution of a model can be employed by a user as a decision support system, an expert system, a job aid to assist

someone in the conduct of a task, to develop detailed text to be employed in traditional interactive courseware or for intelligent tutoring systems curriculum development.

Figures 40a-e present the logic flow diagram for this process.

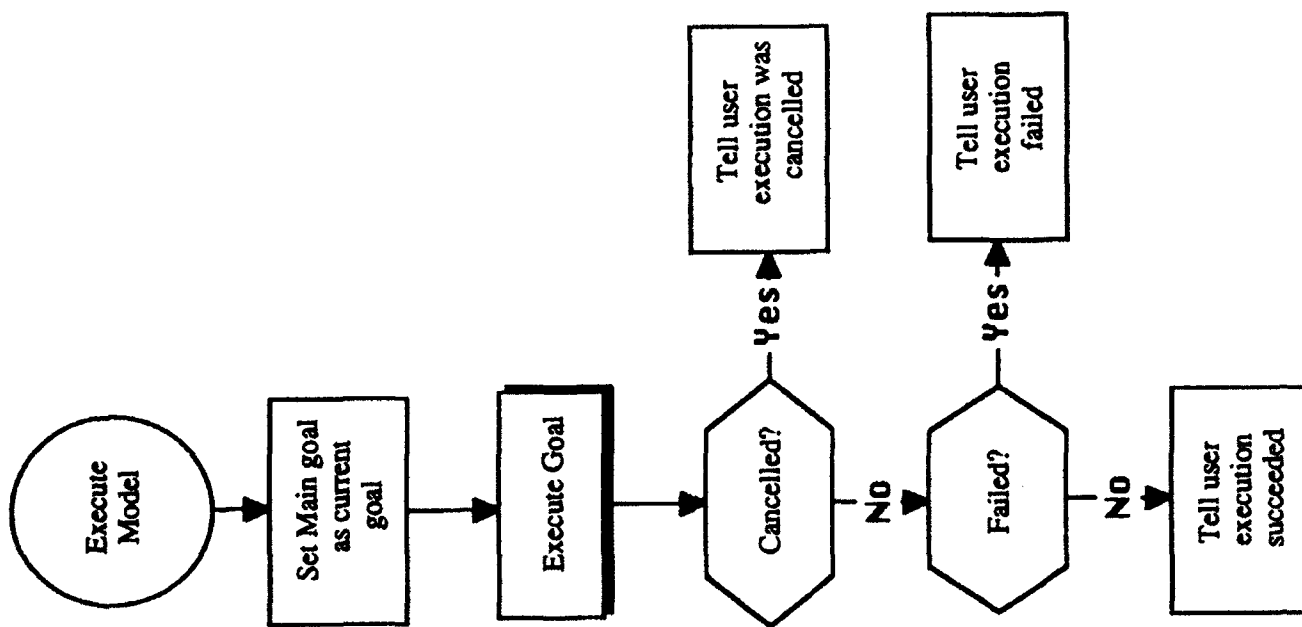
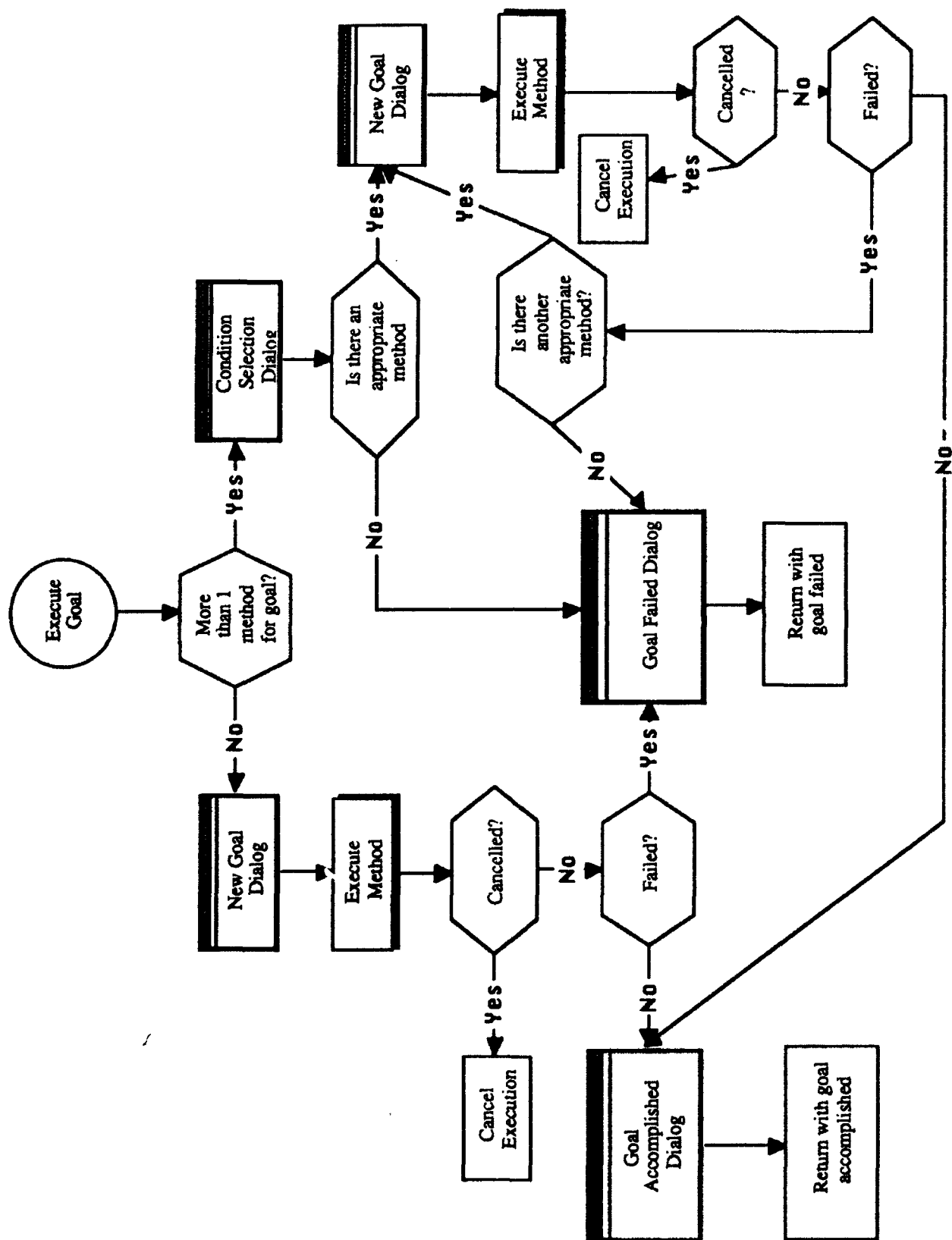


Figure 40a- Logic flow diagram for Model Execution process in CAT



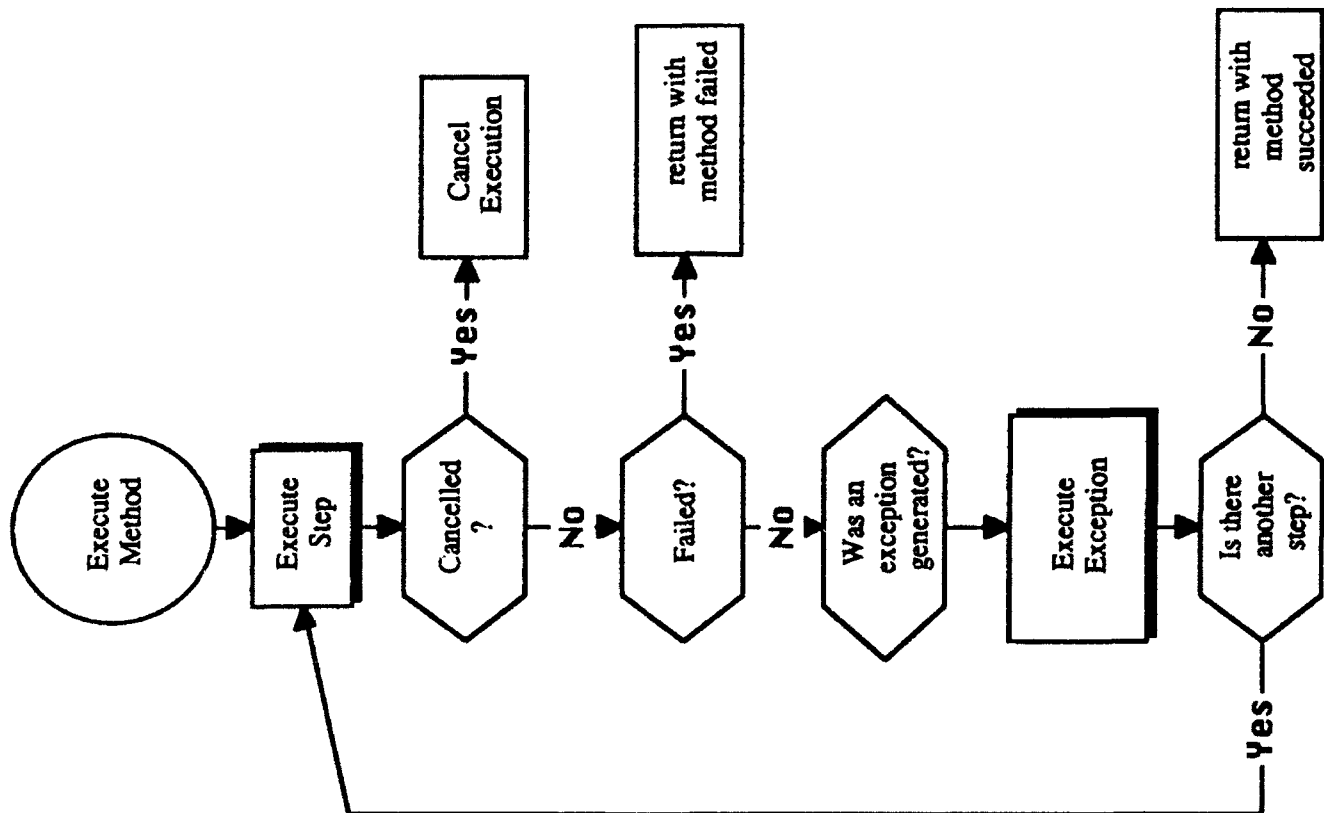


Figure 40c- Logic flow diagram for Method Execution process in CAT

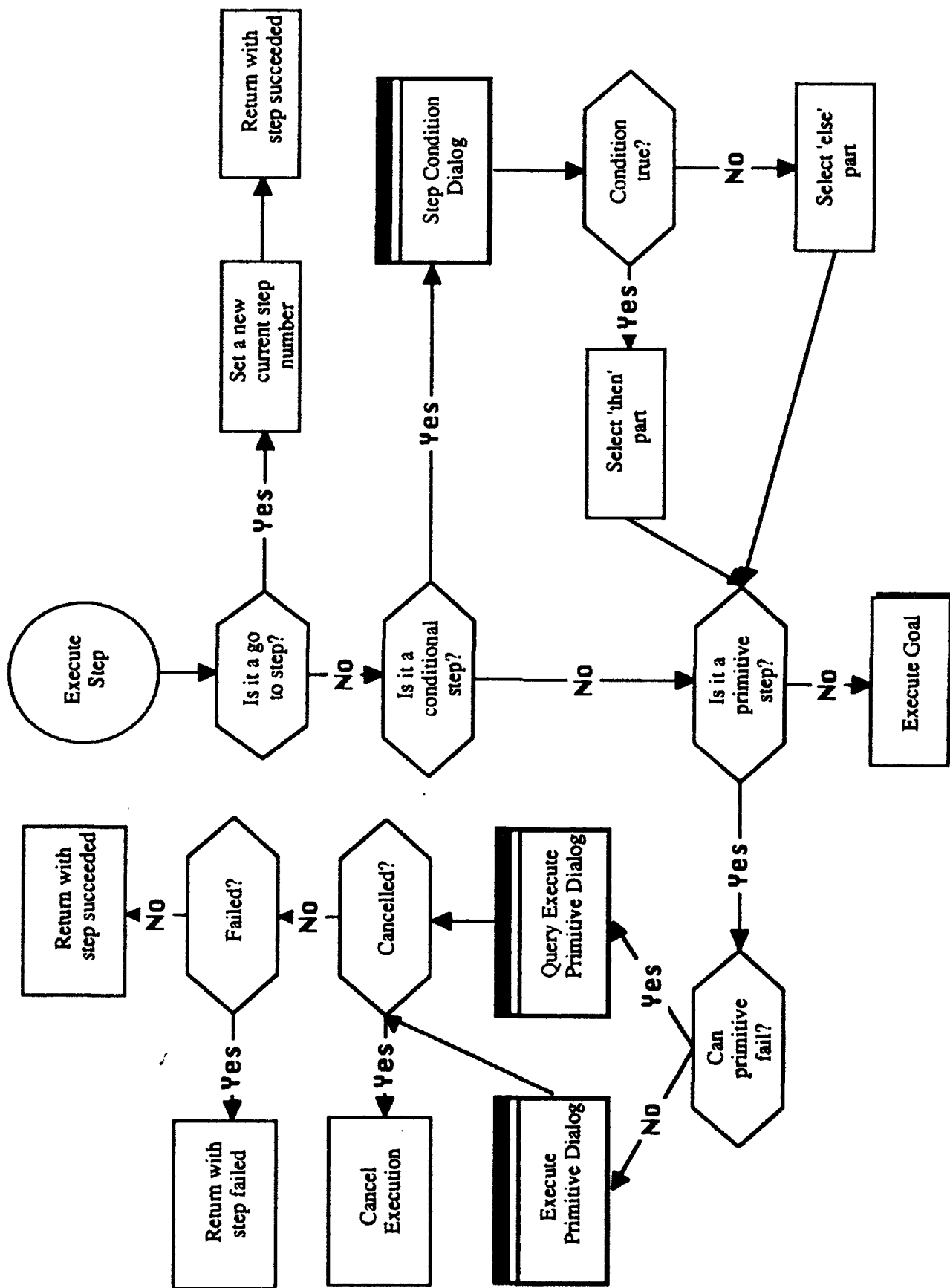


Figure 40d - Logic flow diagram for Step Execution process in CAT

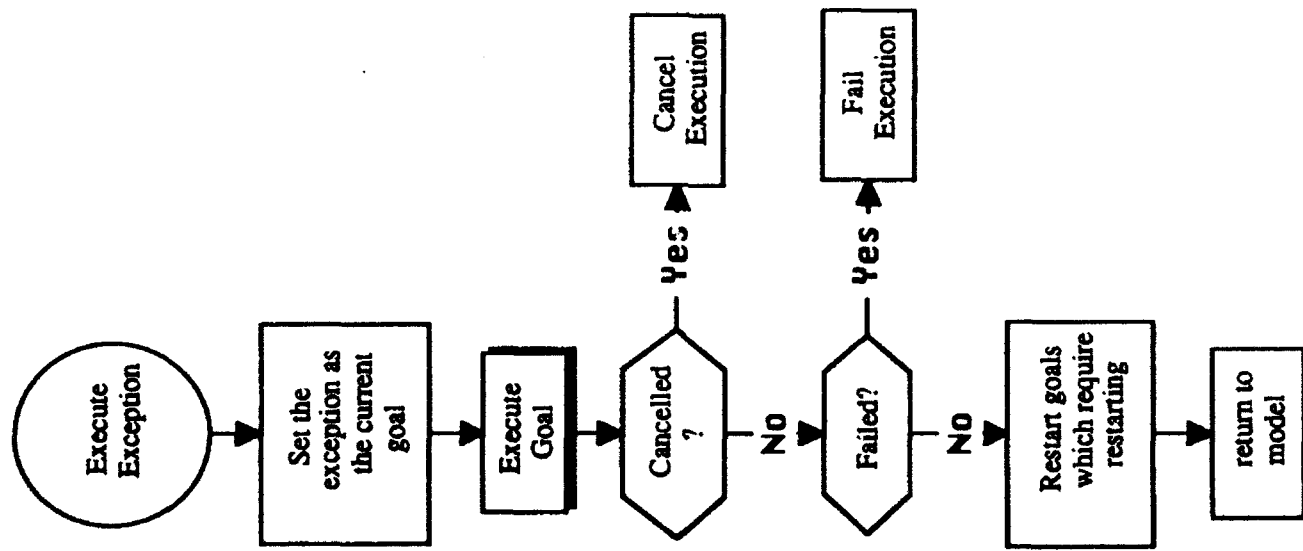


Figure 40e- Logic flow diagram for Exception Execution process in CAT

In order to execute a completed model, the user must choose "Execute Model" from the "Navigate" menu. The main goal of the model to be executed will be set as the current goal to accomplish. The system will check to determine if there is more than one method for this main goal. If not, then the "Execute Goal" dialog box is presented as in Figure 41. If there is more than one method then the Selection Rules Introduction dialog box of Figure 42 is presented, followed by the "Condition Selection" dialog box of Figure 43.

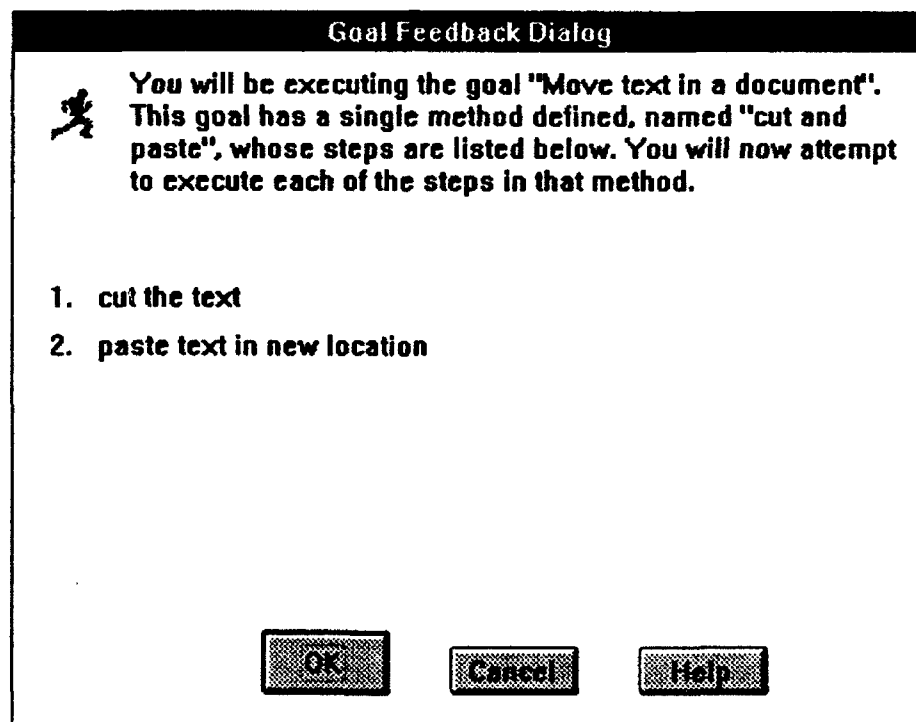


Figure 41- Execute Goal Dialog

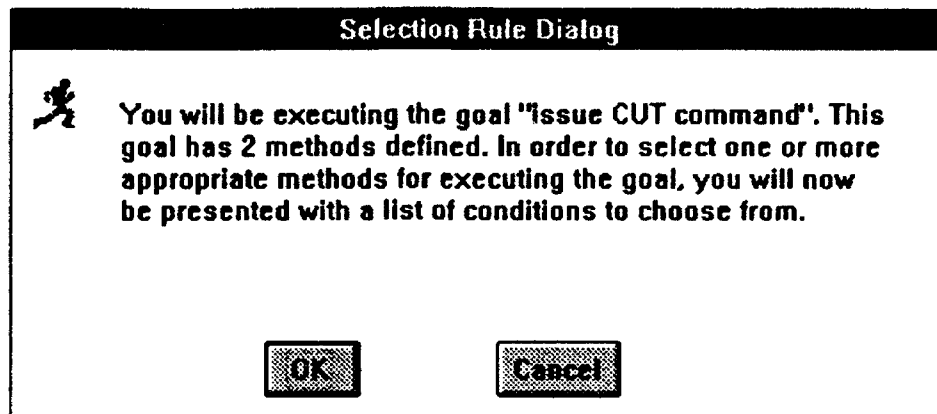


Figure 42 - Selection Rules Introduction Dialog

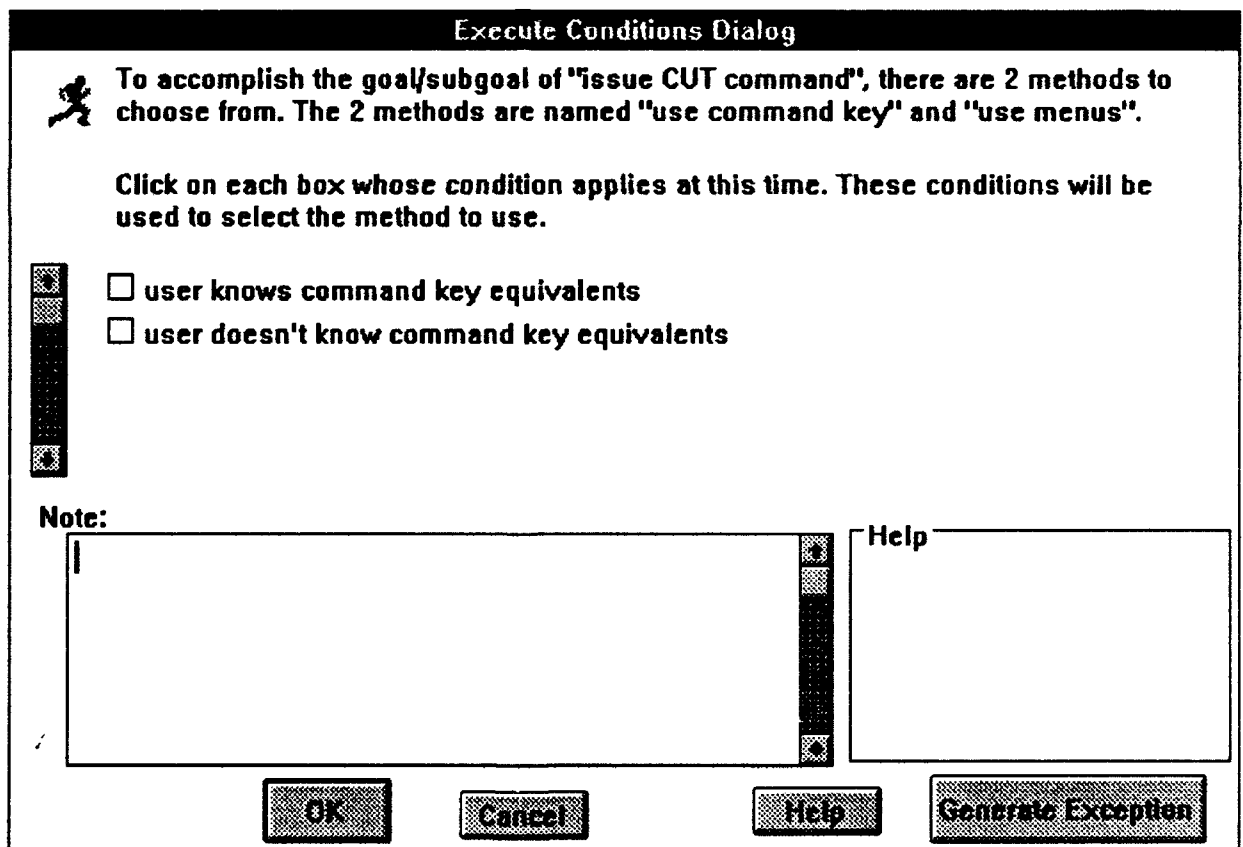


Figure 43 - Condition Selection Dialog

If the "Condition Selection" dialog box is displayed the user is requested to check the appropriate conditions which apply at the current time. The system will then search for the appropriate method associated with these conditions.

During the execution of a Selection Rule, the user may also generate exception conditions by selecting the Generate Exception button of this dialog box. This button will be available whenever the user is selecting Selection Rule conditions, whenever a goal or a step was failed and whenever a primitive step is being executed. If a selection rule is not triggered the "Goal Failed" dialog box depicted in Figure 44 is presented. If on the other hand a selection rule is fired the user will be presented with the "Execute Method" dialog box of Figure 45 and instructed to execute the method. If the execution of the method fails the system will check for another selection rule whose conditions were checked in the "Condition Selection" dialog box. If another method can be triggered then the "Method Failed" dialog box of Figure 46 is presented, followed by the Execute Goal dialog box of Figure 41 in preparation for the execution of a different method. If another appropriate method cannot be found then the system will present the "Goal Failed" dialog box. On the other hand if a method executes successfully then the "Goal Accomplished" dialog box of Figure 47 is displayed to the user.

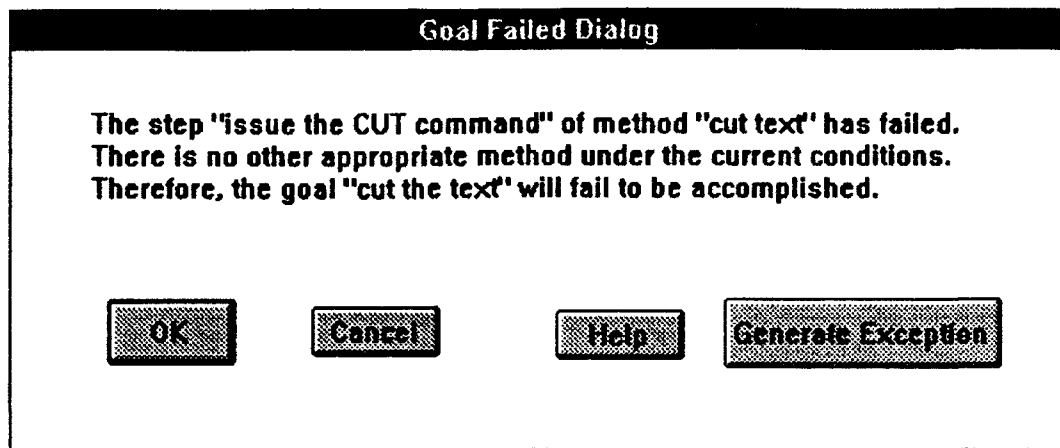


Figure 44 - Goal Failed Dialog

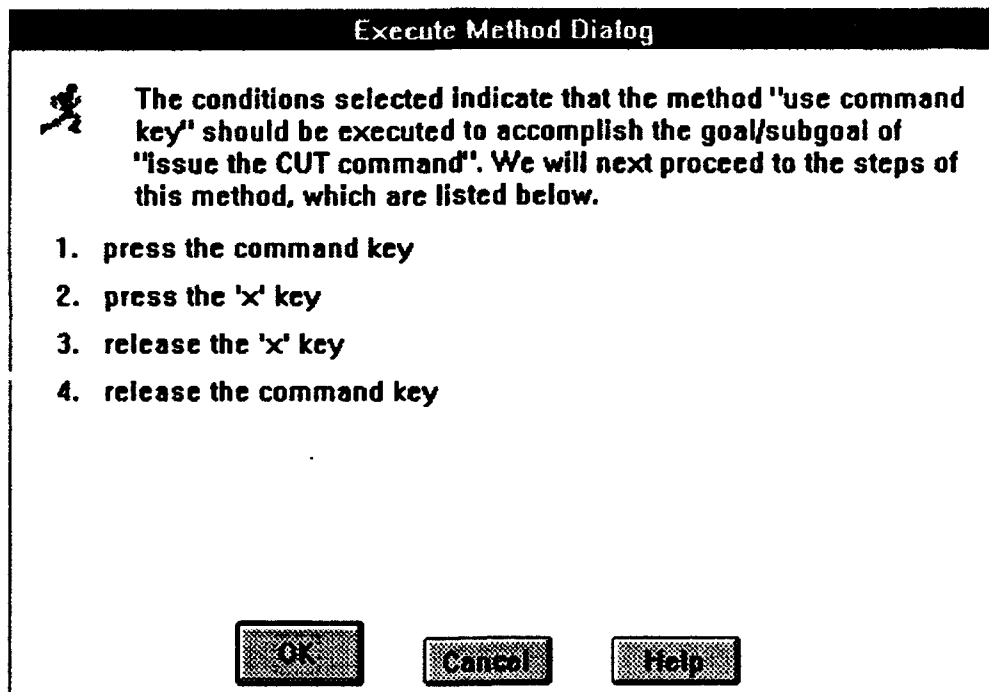


Figure 45 - Execute Method Dialog

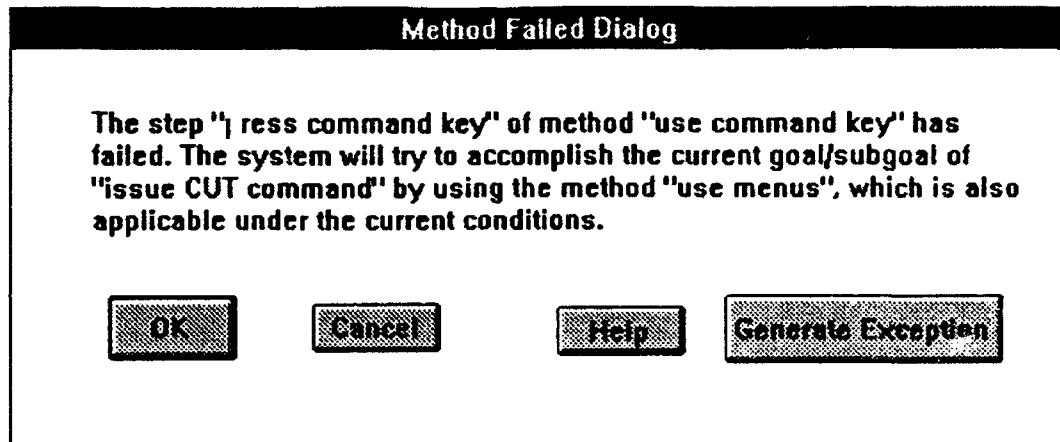


Figure 46 - Method Failed Dialog

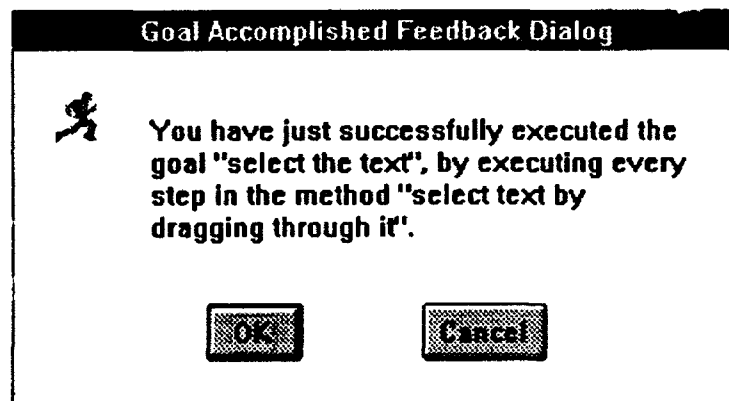


Figure 47 - Goal Accomplished Dialog

When a method is executed, each step is tested to determine if the step involves a GO TO and if the step involves an IF-THEN-ELSE decision structure. If the step is a GO TO, the system sets a new step as the current step to be executed. If the step involves a set of conditionals then the user is presented with the "Step Condition" dialog box of

Figure 48 and required to respond to the truth of the conditions presented. If the conditions are true the THEN part of the decision structure is fired and the step is checked to determine if it is a primitive. If it is not a primitive then a new goal is set for execution. If the truth of the conditionals is not satisfied then the ELSE part of the decision structure is fired and its step is tested to determine if it is a primitive. If not a new goal is set for execution.

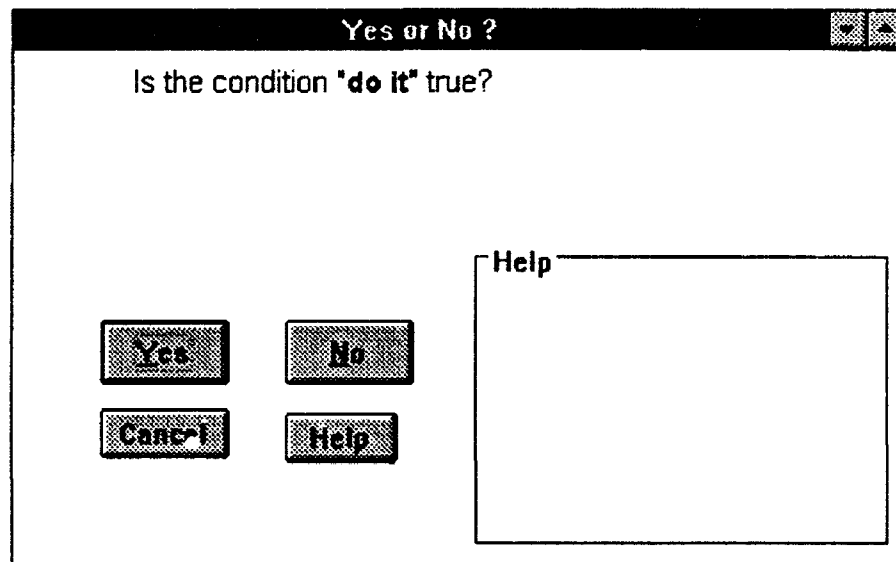


Figure 48 - Step Condition Dialog

If any step being executed is a primitive step then the system checks to determine if the step defined in the model can fail. If the step cannot fail then the Execute Primitive dialog box of Figure 49 is presented. If the step can fail then the "Query Execute Primitive" dialog box is presented as in Figure 50. Depending upon the user's response to these dialog boxes the system will return a failed step or a successful step. If the step fails the system will return a failed method or goal. If a primitive level step succeeded the

system will then attempt to determine if any exception conditions were generated. If not the system will select the next step in the method for execution.

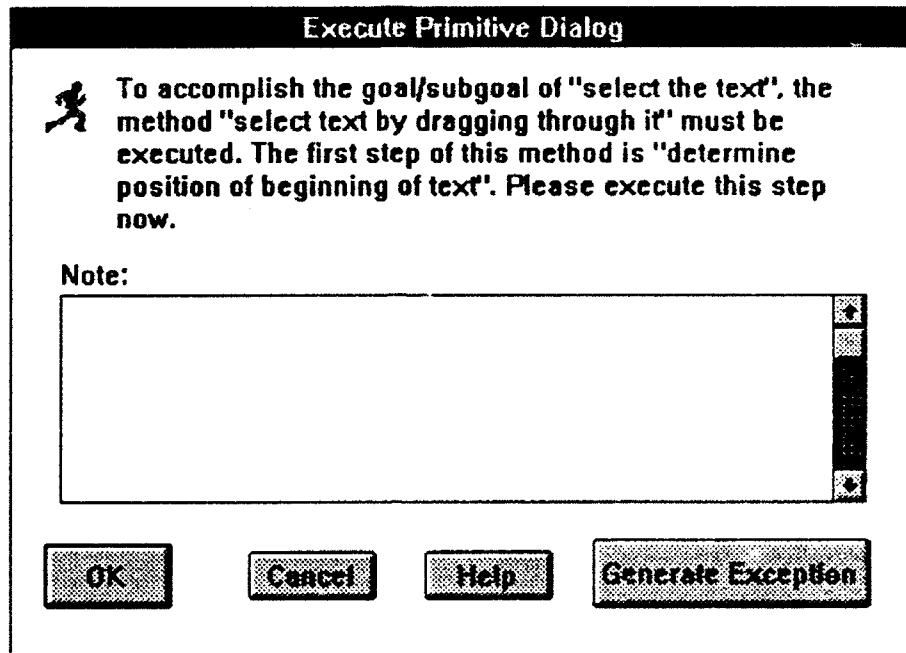


Figure 49 - Execute Primitive Dialog

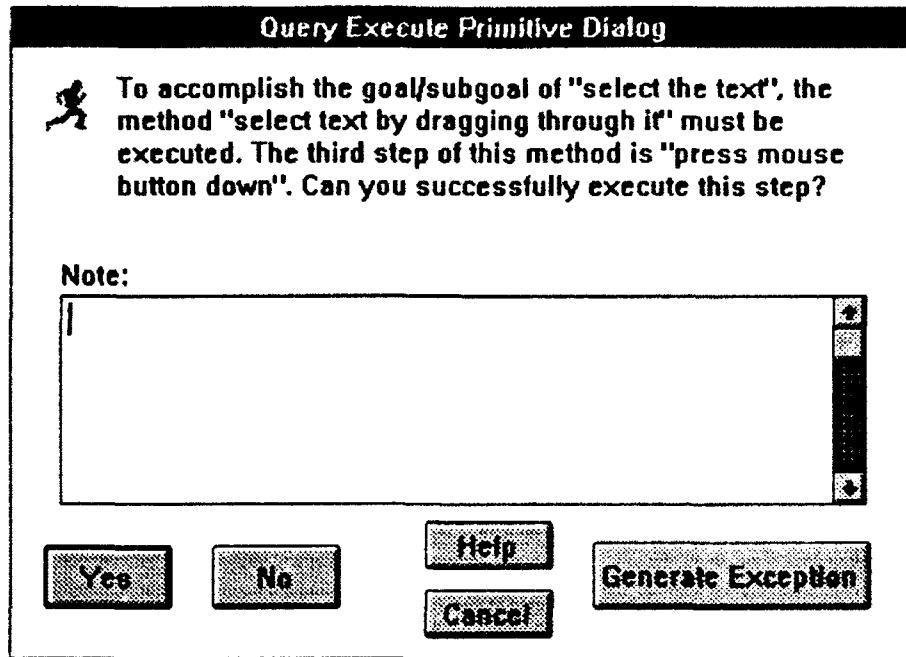


Figure 50 - Query Execute Primitive Dialog

If exception conditions are generated by the user, the system will set the goal associated with the exception rule as the current goal to be accomplished and will attempt execution as for any other method. To determine which exception to generate, the Execute Exception Dialog box depicted in Figure 51 is presented to the user. This dialog box allows the user to specify which conditions are currently in effect, and depending on the user's response, an appropriate exception is generated. Next, the user is presented with the Execute Exception Rule Dialog box of figure 52, which informs him/her of the exception being generated. When the exception rule goal has been accomplished, the user is presented with the Exception Rule Accomplished dialog box of Figure 53, and the system will restart all the subgoals required when execution returns to the main model. The system will continue the execution process until a method fails and the goal cannot be accomplished by way of any other alternative methods or until all subgoals succeed.

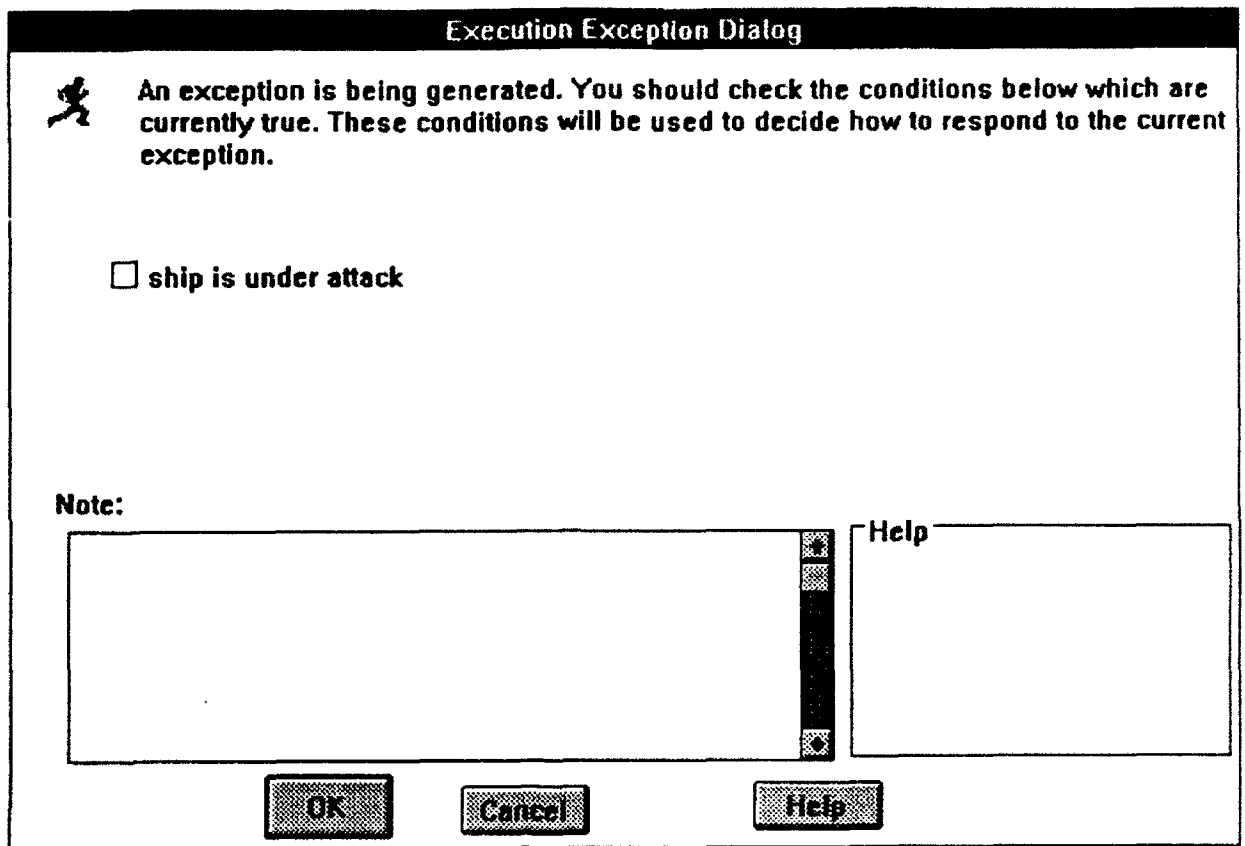


Figure 51 - Execution Exception Dialog

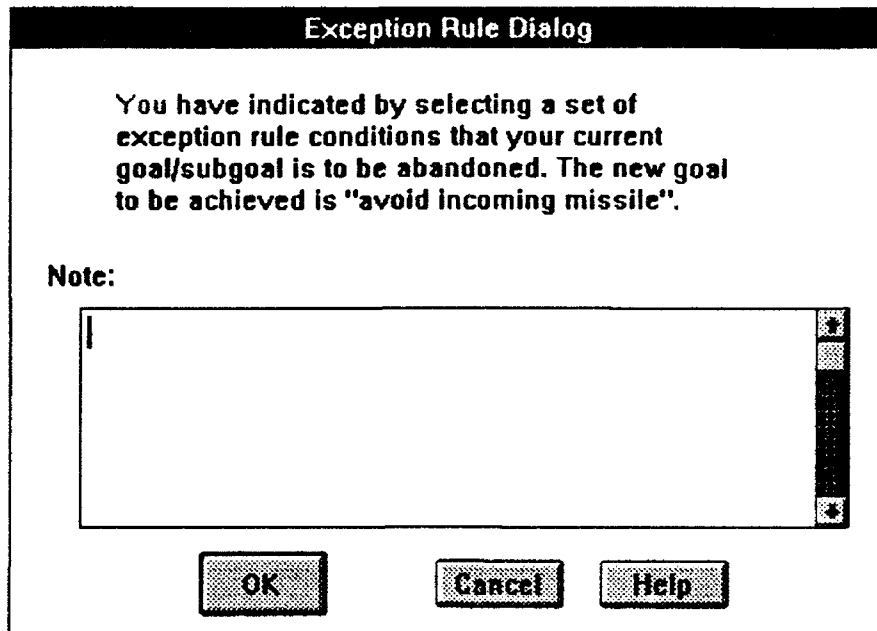


Figure 52 - Execute Exception Rule Dialog

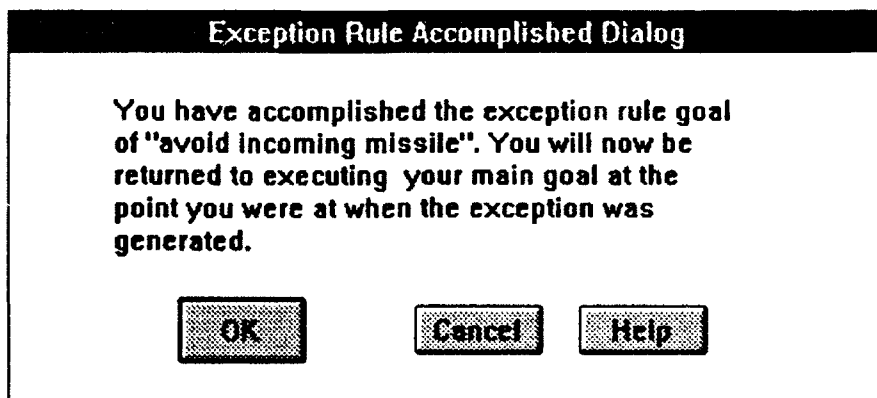


Figure 53 - Exception Rule Accomplished Dialog

## **Formative Evaluation of Modeling Capability**

There were several iterative phases to developing the computerized aid implementing this GOMS analysis process. The first phase included a conceptual analysis, a requirements analysis, system design and system development. The second phase consisted of an evaluation of the aid. This evaluation gave insight into the aid's performance and consequently into how to improve the aid. The information and knowledge gained from this evaluation was then used to iterate through the design cycle and improve the tool. The evaluation phase focused upon investigating the accuracy and consistency with which cognitive models could be generated by using the aid.

The purpose for developing the aid was to assist individuals, naive about cognitive science, in eliciting, organizing and representing their task knowledge in the form and structure of a cognitive task model. Evaluating how well individuals naive to the cognitive task modeling process can represent their task knowledge using the aid can provide insight into improvements to the aid, as well as, to determine to what extent such systems can reduce requirements for cognitive analysts in the modeling process.

Since this research was exploratory development in nature, the current literature did not contain any adequate metrics to evaluate the effectiveness of such a system. The initial evaluation of the system involved a relatively simple well structured task to serve as a baseline from which future improvement and extensions could be made. The questions which needed to be answered by this evaluation were: 1) How accurate and consistent are the cognitive task models developed using this aid, and 2) How could the aid be improved to facilitate the cognitive task modeling process?

## **Methodology**

Subjects: A total of forty two (42) test subjects were selected from a pool of secretaries skilled in word processing within the Department of Industrial and Systems Engineering of Virginia Tech. Of the forty two subjects selected a total of forty (40) were used in the analysis. One subject's data was not used since the subject was under a personal time constraint and did not complete the task. Another subject's data was not used due to a fault in the system during the experimental session. Each test subject was paid five dollars (\$5.00) for participating in the experiment. Each subject participating in the experiment was selected based upon their ability to perform the experimental task.

Experimental Materials and Equipment: Two microcomputers were employed during the conduct of the experiment. An Apple Macintosh with system software 7, a mouse input device and Microsoft Word version 5.0 was used to screen potential participants. The actual experimental task was performed on a WIN 386 IBM PC compatible computer running DOS version 5.0 and Microsoft Windows version 3.0. The WIN 386 was equipped with a mouse input device, a VGA color monitor and eight (8) mb of ram. The automated cognitive analysis tool software was run on this platform during the experimental trials.

Experimental Procedure: The experimental procedure consisted of three stages. The first stage involved screening potential participants to verify the subject's ability to expertly perform the task which was to be described employing the cognitive analysis tool. This stage served as a measure of the test subject's ability to perform the task which was later to be described during the experimental test trial. The subject passed the screening task if he/she could perform each step of the screening task without using the help facility

or the user manual. The subject was therefore required to retrieve knowledge about task performance from memory without any assistance. The task to be performed during the screening test, and to be described during an experimental trial using the CAT modeling system, consisted of a simple editing task involving the Cut and Paste procedures for moving text in the Apple Macintosh environment running Microsoft Word version 5.0. A model adapted from Kieras (1988) describing the task to be modeled by the students is presented in Figure 56. This model is referred to as the baseline model representing a cognitive scientist's analysis of the tool. The instructions to each potential participant including the operations to be performed and the text on which these operations were to be performed appear in Table 1.

If the subject was able to complete each operation described without error, he/she was retained for experimental trials. If not the subject was excused from any further participation and paid one dollar (\$1.00) for his/her effort. If a potential experimental subject was able to complete the screening task without fault, he/she progressed to the second stage of the procedure which consisted of a familiarization of the CAT modeling system.

**Table 1**

**Task Explanation for Screening Session**

During this session you are to perform the following operations on the text displayed on the computer screen.

**FOLLOW THE STEPS EXACTLY AS DEFINED.**

**Operations to be done on the text.**

1. Select the word "out" in line 6.
2. Cut the text with the **cut menu command**.
3. Move the cursor to the bottom of the paragraph and insert the text by using the **paste shortcut/command/keyboard equivalent/quick keys**.
4. Select the text beginning in line 9 with "degrees" and ending in line 10 with "Virginia."
5. Cut the text with the **cut shortcut/command/keyboard equivalent/quick keys**.
6. Move the cursor to line 1.
7. Insert the text with the **paste menu command**.

**Text to Perform Operations on:**

1    **Text Example**

2

3    The College of Engineering has a reputation for offering an excellent  
4    education for the student who desires to obtain a baccalaureate degree.  
5    Today's engineering freshman class has an average scholastic achievement  
6    score of 1200 out of a possible 1600. This is significantly higher than the  
7    mean national score of 906. By graduating more than 1,000 students each  
8    year, the college consistently ranks among the top ten in the number of  
9    baccalaureate degrees granted. The College of Engineering is the second  
10   largest college at Virginia Tech. Graduate and undergraduate enrollment  
11   has increased 44% from 1976 to 1988. The greatest increase has been in  
12   graduate enrollment

13

The familiarization session consisted of three parts. First, an explanation was presented of the CAT modeling system by the experimenter, including definitions of GOMS terminology, as that terminology related to an example problem describing a worked out model of how to "prepare to drive a car", Figure 54. Second, the experimenter worked through the "prepare to drive a car" example using the actual aid. During this process the experimenter explained all of the features of the aid while interactively inputting the model into the system. Third, the experimenter requested the subject to interact with the system. The subject was provided with another demonstration model which addressed how to "prepare an envelope for mailing" (Figure 55) and was instructed to input the model into the system using the features described and demonstrated. During this part of the familiarization, the experimenter provided feedback and answered any questions the subject had about interacting with the aid.

The last stage of the experimental procedure involved having each test subject interact with the CAT modeling system to build a task model describing how to move a piece of text using the Cut and Paste capabilities of Microsoft Word for the Macintosh. Table 2 presents the explanation of this session provided to each subject. In addition to the instructions and explanation presented in Table 2 the experimenter made specific facilitation's to help each subject understand how to use the tool when the subject was having difficulty. If the subject started to define primitive steps for accomplishing the top level goal, the following was said, "You are probably starting at too low a level of detail. Think about the high level steps necessary to accomplish the top level goal. You will be able to define the primitive steps for each high-level goal later on. You can look at the methods in the previous session to help you". If the subject defined two primitive actions per step, then the following was said, "You should define only one primitive action per step. There is a way to define concurrent actions".

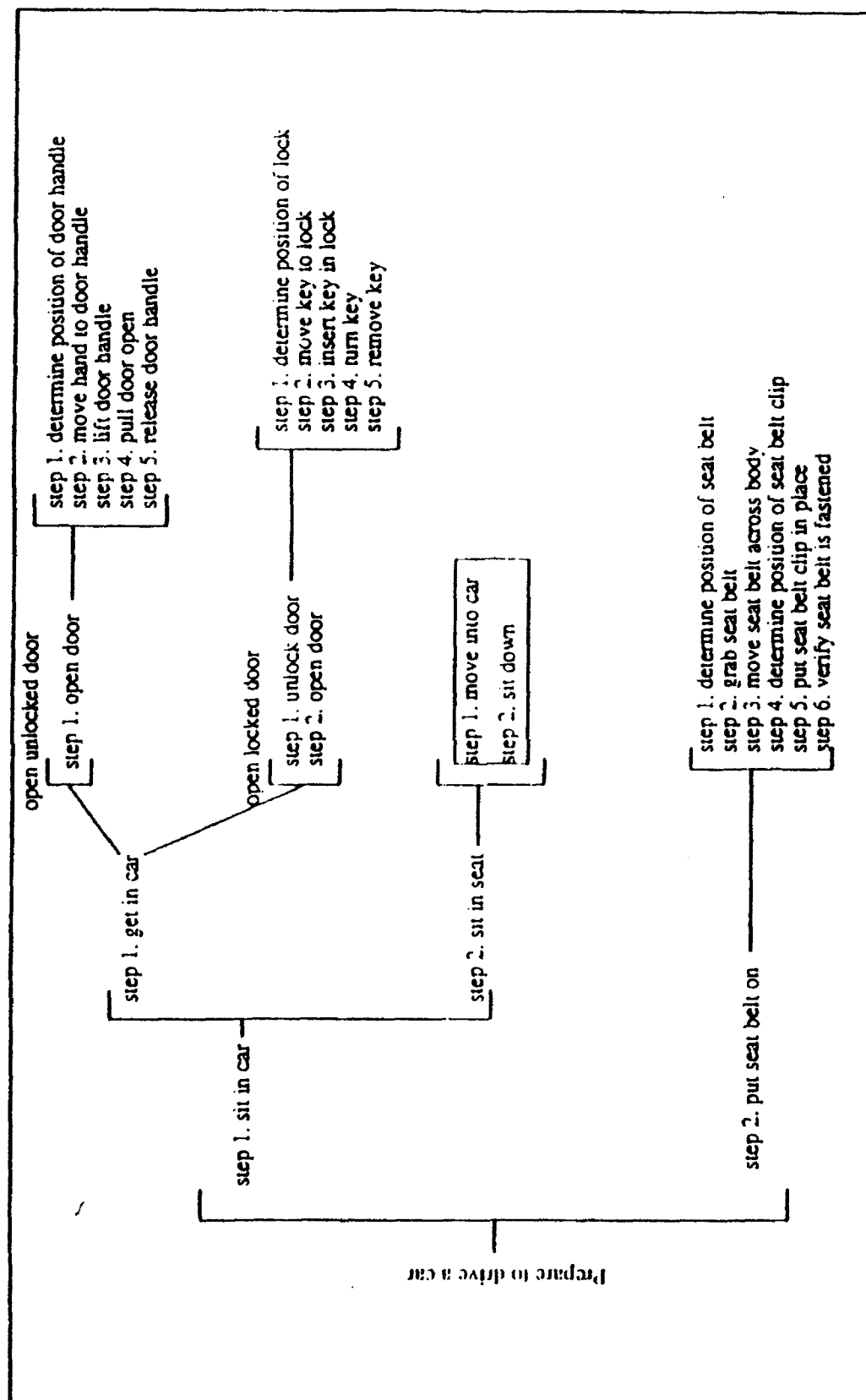


Figure 54: Prepare to drive a car demonstration model. (Note: the steps "move into car" and "sit down" are highlighted by the box to show them as concurrent actions.)

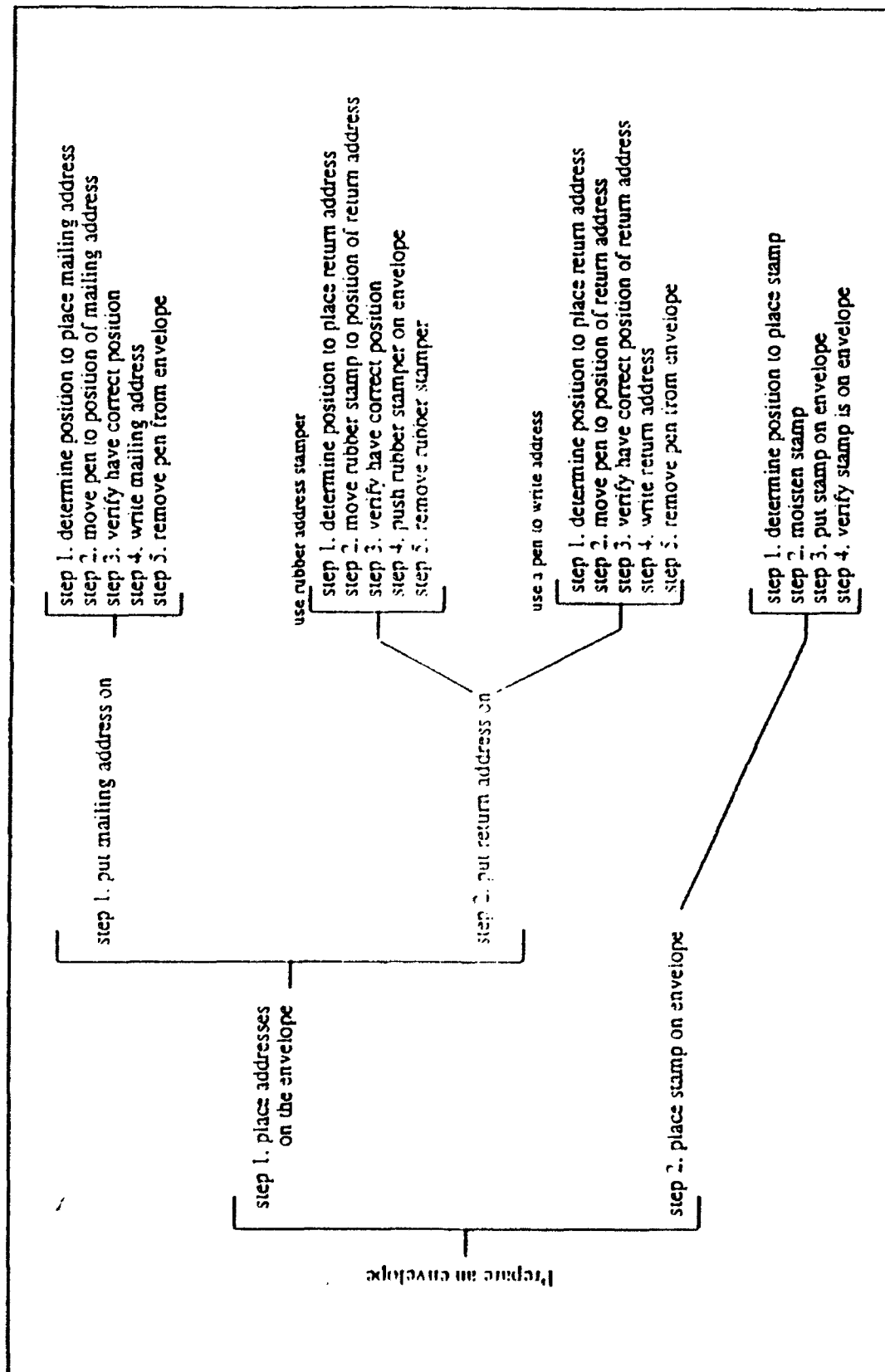


Figure 55: Mail a letter demonstration model.

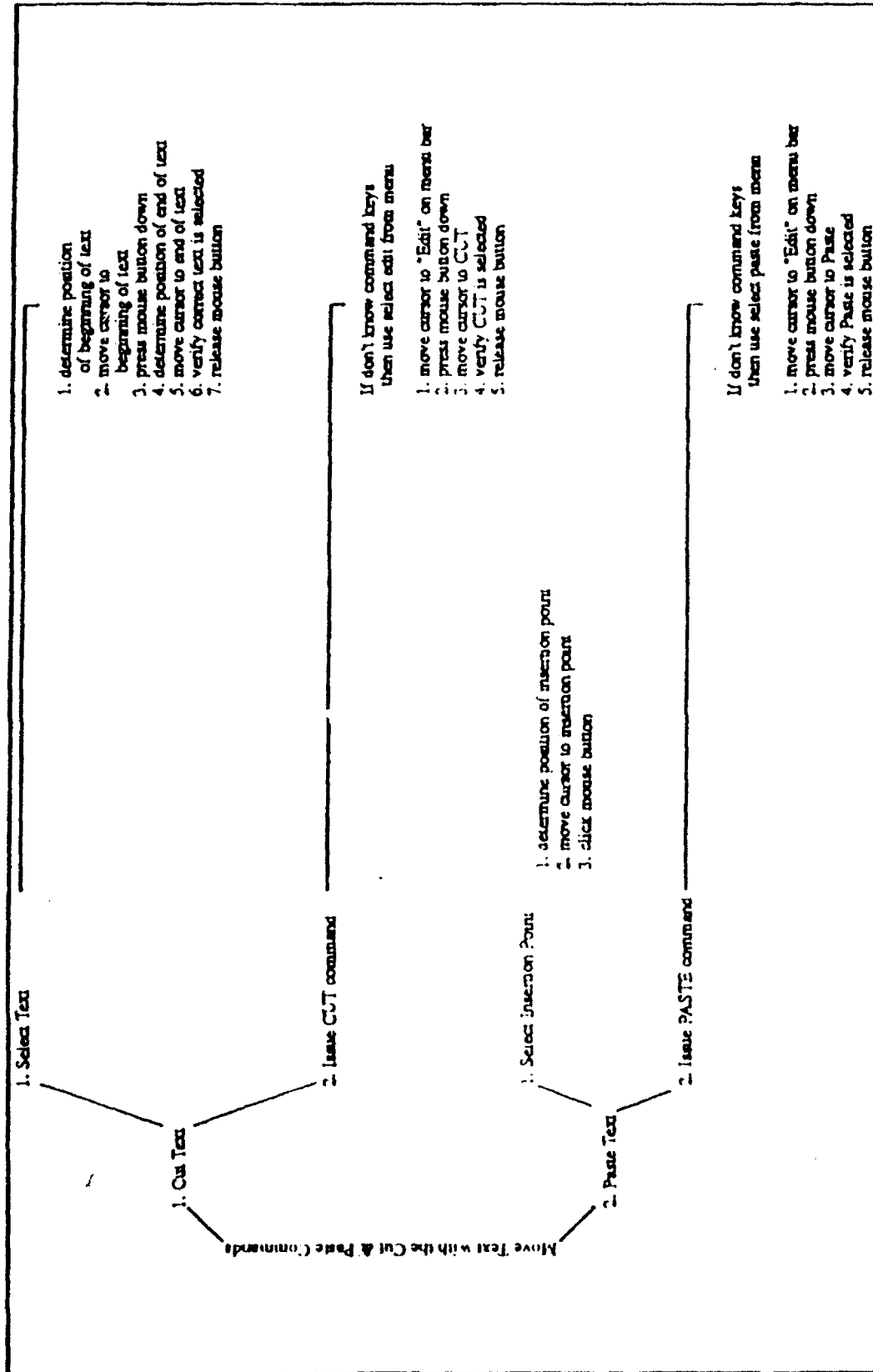


Figure 56: Model of baseline knowledge base for the experimental task. (Adapted from Kieras 1988)

**Table 2**

**Task Explanation for Machine-Aided Session**

During this session you will use a tool called CAT. The tool will prompt you for information about how to accomplish a goal.

Remember to describe both the physical and mental actions necessary to perform the task.

**The goal is:**

Move a piece of text using cut and paste while using Microsoft Word for the Macintosh.

You can begin this session by double-clicking the CAT icon and selecting a new model with the top-level goal of "move a piece of text using cut and paste"

**Assumptions**

When defining the task you can assume the following:

- 1) the Macintosh computer is on,
- 2) Microsoft Word is loaded,
- 3) the text module is loaded, and
- 4) the following are examples of primitives for this session:

**Physical primitives**

moving the cursor  
double-clicking the mouse button  
pressing a key  
releasing a key

**Mental Primitives**

verifying an action.

**Example use of the primitives:**

Goal: move a file to the trash  
step 1. determine position of file  
step 2. move cursor to file  
step 3. press mouse button down  
step 4. verify correct file is selected  
step 5. move cursor to trash

Following completion of the model building process each subject was requested to match each step which they defined with a corresponding step description from the baseline model. This comparison matching was done to ensure that subject-generated semantics were consistent with the names ascribed to each step in the baseline model. Instead of having the experimenter make judgments about semantic similarity, the subjects were asked to make these judgments themselves since they would best understand how the terms they generated were to be interpreted.

Performance Measures: The performance measures for this experiment were designed to assess the capability of the system to elicit accurate and reliable cognitive task models from a user. There were two dependent measures calculated based upon what each subject-generated model was compared to: (1) subject model versus a baseline model and (2) subject model versus other subject models. Comparisons of each subject's model to the baseline model were made to determine a measure of accuracy of the models generated. The baseline model was adapted from Kieras (1988) and is presented in Figure 4. This adaptation removed any "return goal accomplished" statements from the model generated by Kieras since this is automatically inserted by the tool as the last statement of each method defined. Short cut key methods were also excluded in the adaptation of the Kieras model since our pilot study indicated that only a small percentage of users representing the population of secretaries knew these short cut keys. Each high level method and each primitive level method generated by a test subject as a result of model development was compared to this baseline model. Accuracy was calculated as the percent match between subject generated methods and the corresponding baseline methods. The accuracy for a given method was computed by dividing the number of steps of a given baseline model method that a subject defined in his/her corresponding method by the total number of steps described in that baseline model method. This was done for each method of the baseline model (see Table 3).

Comparisons of each subject model versus other subject models were made to determine the consistency of models generated between subjects. Each model generated by each subject was compared to each of the other subject models. Consistency of the models provided a measure of variation in the models generated. If the system is to be effective in generating cognitive task models of domain expertise, it must generate similar models from different expert's on the same well structured tasks. Consistency was calculated for both high-level methods and primitive level methods. Consistency measures were calculated as the percent match between a subject generated step of a given method and the corresponding step in the same method across all subjects. The number of subjects who had the step in a method was counted, as was the number of subjects who did not have the step in their corresponding method. The consistency of a step was calculated by dividing the maximum of either the number of subjects who described the step in the method or the number of subjects who did not describe the step in the given method by the total number of subjects. Method consistency was computed as the average of the step consistencies within each method (see Table 4).

**Table 3**

**Example Calculations for the Accuracy Performance Measure**

<b>Method: Select Insertion Point</b>	<b>Subject</b>			
	<b>#1</b>	<b>#2</b>	<b>#3</b>	<b>#4</b>
<b>Primitive Method Steps</b>				
determine position of insertion point	1	0	0	1
move cursor to insertion position	1	1	1	0
click mouse button	1	0	1	1
# of baseline primitives defined by subject for this given method	3	1	2	2
# of baseline primitives in this given method	3	3	3	3
Accuracy	3/3	1/3	2/3	2/3
Note: Table values: "0" = subject didn't define the primitive in this method, "1" = subject did define the primitive in this method.				

Table 4

Example Calculations for the Consistency Performance Measure

Method: new position	Subject				# of subjects who had step for this method	# of subjects who did not have step for this method	Subject Generated Primitive Method Consistency
Primitive Method Steps	#1	#2	#3	#4			
determine position of insertion point	1	0	0	1	2	2	.500
move cursor to insertion point	1	1	1	0	3	1	.750
click mouse button	1	0	1	1	3	1	.750
verify have correct position	0	1	0	0	1	3	.750
Note: Table values: "0" = subject didn't define the primitive in this method, "1" = subject did define the primitive in this method.							

## Experimental Analysis

The final step in the data analysis process consisted of gathering all of the methods generated by each test subject. In reference to the baseline model, there were seven (7) methods. These seven methods consisted of three (3) high level methods, one which served the highest-level subgoal of "Move Text with the Cut and Paste Command" and the others which served the two subgoals of "Cut text" and Paste text". The remaining four (4) methods were identified as primitive methods; one for the "Select Text" subgoal, one for the "Cut Text" subgoal, one for the subgoal of "Select Insertion Point" and one for the subgoal of "Paste Text". A  $2 \times 4$  repeated measures analysis of variance (ANOVA) was conducted on accuracy scores calculated for the four primitive methods defined in the baseline model. This analysis partitioned the primitive steps in each method into physical primitives and mental primitives. This allowed for a comparison of accuracy of models in terms of the percentage of mental primitives reported versus the percentage of physical primitives reported. Additionally, any differences between the accuracy of descriptions for any given primitive method could be assessed with this analysis scheme.

The analysis of method consistency involved the comparison of each primitive method generated by each subject to the corresponding primitive methods of each of the other test subjects. As a result, measures of the means and standard deviations for each method across all 40 test subjects were computed. The mean and standard deviation scores of the percent match between steps of a method yields an indication of the degree of variation in the subject generated models.

## Results

The initial intent to analyze the accuracy and consistency of both the high level and low level methods generated by the test subjects proved to be quite challenging. An initial assumption in this experimental evaluation was that subjects would demonstrate a high degree of accuracy and consistency in the models generated since the task was highly structured. The analysis of the goal/subgoal hierarchies which reflects how the methods were mentally organized by each test subject demonstrated a considerable lack of accuracy when compared to the baseline goal/subgoal hierarchy. Only three (3) subjects of the 40 who participated in the experiment generated models whose goal hierarchies matched that of the baseline model. Of the remaining 37 models generated, a total of 20 different goal/subgoal hierarchies were identified. As a result of this lack of match between test subject hierarchies among themselves, and in comparison to the baseline model, a closer inspection of the goal/subgoal trees was made. As a result of this examination, it was found that many of the subject generated models consisted of primitive methods which were either a composition of two baseline model primitive methods or a decomposition of a single baseline model primitive method into two subject generated primitive methods. Composition of production units is a well documented phenomenon relative to the process of knowledge compilation (Neves and Anderson, 1983 and Anderson, Greeno, Kline and Neves, 1983). Composition occurs during skill development when two production units, or rules which typically are executed in sequence are combined into a single production unit. Consequently, what we are seeing in this examination of test subject primitive methods, is a variation in the manner in which they compose their knowledge about the task, relative to the baseline model primitive level methods. Fourteen (14) subjects composed primitive level methods differently than those methods reflected in the baseline model. That is, of the 40 test subjects in the experiment, fourteen demonstrated instances

when two methods of the baseline model were composed into one method in the subject generated model. Of the 40 test subjects in the experiment, 20 subjects demonstrated instances where one method of the baseline model was decomposed into two methods in the subject generated models. An example of two subject generated methods which if composed would form a single baseline method is described in the following:

#### Two Subject Generated Methods

GOAL "cut text with menu"

- 1). "move cursor to edit"
- 2). "press mouse button"
- 3). "select cut"

and

GOAL "select cut"

- 1). "move cursor to cut"
- 2). "release mouse button"

#### Baseline Method

GOAL "cut text with menu"

- 1). "move cursor to edit"
- 2). "press mouse button"
- 3). "move cursor to cut"
- 4). "release mouse button"

In other cases a test subject may have described a single method composed from two baseline methods. For all practical purposes, there is no difference between a composed rule and the two production rules from which it is composed in terms of the model description correctly reflecting how to do something. Consequently, the primitive methods generated by each test subject were either composed or decomposed such that they would reflect greater agreement with the baseline model where it was evident that a subject either composed or didn't compose their methods in agreement with the baseline model. When these adjustments were made to the primitive level methods described by the test subjects, the degree of variation in the goal/subgoal hierarchies of test subject

models as compared to the baseline goal/subgoal hierarchy was dramatically reduced. The resultant number of different trees generated was reduced from twenty (20) to three (3).

### **ANOVA on Accuracy Measures**

The 2 x 4 repeated measures analysis of variance conducted on the percent accuracy scores calculated on the four primitive level methods of the model revealed a significant main effect of type of primitive (i.e., mental versus physical) with  $F(1,39) = 238.69$ ,  $p < .0001$ . This significant difference in the accuracy of the type of primitive step elicited from users indicated a high degree of accuracy for physical primitives with a mean of 85% versus mental primitives with a substantially lower mean value of 28%. Corresponding standard deviation scores for physical and mental primitives are .29 and .40 respectively, indicating less variation in the ability of subjects to describe physical primitives than mental primitives.

A significant main effect for the type of method described by students was also found for measures of accuracy  $F(3,117) = 10.96$ ,  $p < .0001$ . A Newman-Keuls test for critical differences between the mean accuracy scores of each method revealed significant differences at  $p < .05$  between the Paste method and Select Text method (respectively .46 versus .64) between the Paste method and the Insertion Point method (respectively .46 versus .67) and between the Cut method and the Insertion Point method (respectively .50 versus .67). These differences are primarily due to the magnitude of variation between methods in terms of the percent correct mental primitive steps described by the subjects. Cut method and Paste method demonstrated dramatically low scores on the accuracy with which mental primitives were described by test subjects as is demonstrated by a significant primitive type by method type interaction  $F(3,117) = 42.65$ ,  $p < .0001$ . Figure 57 depicts

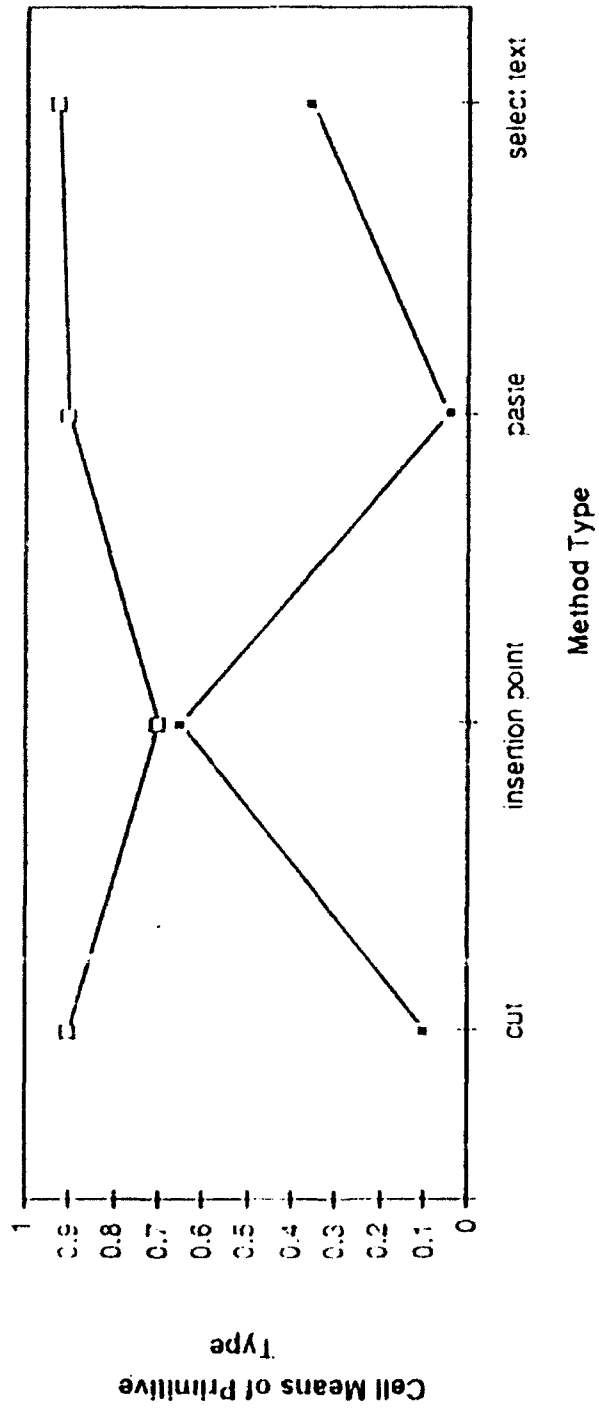


Figure 57: Interaction of Method Type by Primitive Step Type as Percent Accuracy.

the nature of this interaction. The Cut method and Paste method both require a "verify" mental operator in their baseline model descriptions. Whereas Select Text method and Insertion Point method require "determine position" mental operators in their baseline model descriptions. Apparently a "verify" operator is a much more implicit step than a "determine position" operator, even though the step of verifying an action was explicitly identified as a mental primitive in the instructions to test subjects. Verify operators follow a movement of the cursor to a location on the screen, the verification of this movement is simultaneous to the movement and may not be interpreted as a discrete step by non-cognitive analysts. On the other hand, determining a position to begin an action is apparently a more discrete action of which subjects are aware.

### **Analysis of Consistency**

The consistency analysis is intended to provide data relevant to the degree of variation between subject generated models and is therefore descriptive in nature. This analysis provides much needed information regarding the distribution of cognitive task models one might expect among a population of domain experts modeling a well structured procedural task. This type of analysis also attests to the degree of specificity with which one can make predictions regarding the structure of knowledge of domain experts.

The analysis of consistency for the models generated was conducted on the methods adjusted for composition as was the case for the analysis of accuracy. However, for the consistency analysis each subject's methods were compared to every other subject's methods without reference to the baseline model. The means and standard deviations for the consistency analysis are presented in Table 5. The table presents the consistency for

**Table 5**  
**Summary of Consistency Results of Subject-Versus-Subject Comparison**

<b>Method</b>	<b>Primitive Type</b>	<b>Average</b>	<b>Standard Deviation</b>
Select Text	Mental	0.872	0.096
	Physical	0.930	0.057
	Both	0.894	0.085
Cut Text with Menu	Mental	0.922	0.070
	Physical	0.910	0.034
	Both	0.917	0.057
New Position/Insertion Point	Mental	0.750	0.141
	Physical	0.783	0.191
	Both	0.770	0.154
Paste Text with Menu	Mental	0.925	0.071
	Physical	0.905	0.027
	Both	0.917	0.057
Across all Methods	Mental	0.898	0.091
	Physical	0.893	0.090
	Both	0.896	0.090

each method generated by the test subjects. Overall consistency of a method across all 40 test subjects is presented in the row entitled "both". The rows entitled "Mental" and "Physical" partition each method into means and standard deviations for the mental and physical primitive steps of each method. The last row of the table presents the average consistency measures computed across all methods for all test subjects. As can be seen from the table the percent match between subjects across all methods is quite high approaching 90 percent for both mental and physical primitives with the exception of the "Insertion Point" method which demonstrated an average consistency of 77 percent for the combined mental and physical primitives. An examination of the raw data for this method indicated lower levels of consistency in terms of the absence or presence of the "click mouse" physical primitive. Only 50 percent of the test subjects included the primitive step of "click mouse" in their descriptions of this method. The low consistency measure for this step relative to other steps, reduced the consistency scores for this method. The larger standard deviations for the "Insertion Point" method also reflects the lack of consistency in the manner in which test subjects described this method relative to other methods in their models. It is speculated that this method is a rather implicit method and is assumed to be understood when subjects describe their models. In fact, this was the only method included in the baseline model of the task which was not defined by several subjects.

The standard deviations of the remaining methods in the analysis of consistency are quite small attesting to the relative lack of variation in the manner in which test subjects describe their task representation relative to each other. Reference to Table 6 also indicates the narrow bands for the confidence limits on the consistency with which primitive operators are reported or not reported depending upon their type. It must be noted again that the high consistency scores for the mental primitive steps of the models reflect the consistent absence of mention of mental primitives across test subjects.

**Table 6**  
**Confidence Intervals for Performance Measures**

<b>Measure</b>	<b>Confidence Level (%)</b>	<b>Lower Limit</b>	<b>Upper Limit</b>
Physical Consistency	90	0.834	0.908
	95	0.826	0.916
	99	0.810	0.931
Mental Consistency	90	0.869	0.918
	95	0.864	0.923
	99	0.854	0.934
Both Consistency	90	0.861	0.904
	95	0.857	0.909
	99	0.849	0.917
Physical Accuracy	90	0.780	0.874
	95	0.770	0.884
	99	0.903	0.751
Mental Accuracy	90	0.228	0.339
	95	0.217	0.350
	99	0.194	0.372
Both Accuracy	90	0.683	0.773
	95	0.674	0.782
	99	0.656	0.800

Table 6 also depicts relatively narrow bands on the confidence intervals for accuracy of primitive steps described, again attesting to the relative lack of variation one can expect from expert descriptions when developing such models. Consequently, we can be highly confident within a narrow range that experts generating GOMS task models will with a high probability, accurately and consistently describe physical operators. On the other hand, we can be confident that experts generating GOMS task models with this computerized aid will not accurately describe mental operators and will consistently omit them in their models.

### **Discussion and Conclusion**

The objective of this research was to develop a computerized aid to assist subject matter experts in the conduct of a cognitive task analysis employing a GOMS-like process to structure their task knowledge. This knowledge in turn can be used to develop instructional curriculum and ideal student models for intelligent tutoring. The tutoring systems which employ these ideal student models and which structure curriculum in accordance with the structure of knowledge specified by a cognitive task analysis process have been quite successful. (Anderson et al, 1989; Williams et al, 1990). To this end, the results of the design, development and formative evaluation activities of this research have demonstrated that such an aid can accomplish the objective of structuring knowledge about a task domain in accordance with this GOMS-like cognitive task analysis process.

The results must, however, be qualified regarding the level of detail to which non cognitive analysts can use the tool in generating GOMS analyses. As the evaluation has demonstrated, primitive operators associated with mental activities (i.e., implicit cognitive and perceptual operations) are not typically described by subject matter experts. As the results have demonstrated only 28% of the primitive mental operators of a given method

were accurately described in subject generated models. On the other hand, 85% of the physical operators were accurately described in subject-generated models. This result limits the applicability of the aid in its current version to developing models whose content is restricted to descriptions of the rather explicit operations involved in skilled performance. However, this restriction only applies to those not skilled in cognitive analysis. The tool can be readily used by cognitive analysts in developing cognitive task models. Moreover, for purposes of developing and structuring curriculum for intelligent tutors, the models generated by the current version of the aid are sufficient. The presence of mental operator descriptions in the content of curriculum is relatively scant since many of these operations are implicit internal operations which simulate state changes in working memory. The goal hierarchies and selection rules, provide a considerable chunk of cognition and context knowledge to the models currently generated. The models generated are accordingly more cognitive in form and structure than what can be achieved from a traditional behavioral task analysis. Traditional behavioral task analysis focuses upon descriptions of the behavioral steps required to do a task. While a cognitive task analysis also provides considerable description of behavioral events, these events are structured in accordance with what is known about production memory structures. These structures have limitations in terms of the number of steps which can be chained together in a sequence, as well as, conditions which indicate when or under what conditions the behavior is to be executed. Moreover, the control of flow of behavioral sequences is also well specified in a cognitive task analysis as a result of the linkages between the various goals and subgoals which are accomplished by certain behaviors, and the selection rules specified. This control of flow of behavior represents plan knowledge and strategic knowledge (i.e., what-to-do and when-to-do-it knowledge) which is not specified in a traditional behavioral task analysis. Additionally, the output of a cognitive task analysis is executable as a computer program, attesting to the systematicity and specificity with

which such an analysis is conducted, vice a behavioral task analysis. Therefore, although the current version of the computerized aid may superficially appear to be a variant of a behavioral task analysis process, there are considerable differences between this GOMS-like process and a traditional behavioral task analysis in the content, structure and form of its output. It is this structure and systematicity which produces the dramatic effects demonstrated by intelligent tutors over traditional CAI, based upon the output of behavioral task analysis. Consequently, for purposes of developing ideal student models and specifying the structure and content of curriculum, the computerized aid will have a significant impact on the efficiency with which intelligent tutoring systems can be developed.

Of primary importance to the application of intelligent tutoring systems, this aid can considerably reduce the cost associated with the conduct of a cognitive task analysis. The aid will facilitate the accessibility of ITS technology by assisting developers in the complex process of building cognitive task models. As a consequence, the aid would lead to a reduction in the cost to develop ITS and an increase in its breadth of application.

### **Implications for HCI Complexity Predictions**

From a basic research perspective, there has been no known attempt to establish statistical parameters concerning the degree of variation one can expect between the organization of cognitive task models generated by a large sample of experts. Consequently, the analysis conducted, from the evaluation of this computerized aid will add a much needed statistical basis to the domain of cognitive task modeling. As was indicated by the results, consistency in models generated was quite high after the methods generated by the experts were adjusted to a baseline standard. It was quite surprising to

find the degree of variation in methods described, especially for such a well-structured task. GOMS models have traditionally been used to make predictions about device interaction complexity. The complexity of a device in part being a function of the time it takes to execute a method using the device. This variation in the manner in which experts compose their knowledge about a task domain will have an impact upon the variation one might expect when predicting task execution time using GOMS models. For example, one expert may model a sequence of actions to attain a goal as a single method or rule. Another may model a sequence of actions to obtain the same goal as two or three methods each with its own subgoal. The models are both correct. Yet predictions of execution time for these two models will vary although they both specify the same number of steps. The prediction of execution time takes into account the cycle time to set each subgoal. Consequently, the execution time predicted for a goal which requires two methods will be larger than that which contains only one method, even though the models both accomplish the same task correctly. Therefore, in predicting execution time from GOMS models of expertise, there will be a source of variation resulting from the way in which the expert composes his knowledge about a task. This variation is in addition to that which is associated with the cycle times of the various types of operators. Such variation should be taken into consideration when comparing two devices using GOMS keystroke level models. This may especially be true when the differences in execution time between two devices is marginal. It may just be that the differences predicted are the result of how the experts have structured their knowledge of the methods used to interact with the device and not due to a true difference in device complexity. For a given device and a given task, predictions from different expert generated models will show variation as a result of the differing number of methods they described in their models of the same interaction. One could therefore convert expert models to novice representations of the task, since novice representations do not assume any composition of steps in a method. That is, each step of

a method is treated as a single production rule. Reducing expert models to novice level representations should then normalize the variation between expert models and provide more accurate predictions of execution time when comparing two device models. This is the strategy employed when predicting training time from cognitive task models.

### **Predicting Time to Learn and Degree of Training Transfer By Extending CAT**

The modeling capability of the current version of CAT can be extended to produce lower level production rules. These lower level production rules can be used to compute and predict the time it will take a novice to learn a task which has been modeled, as well as, to determine the degree of transfer which can take place between two tasks. The current version of CAT generates methods which reflect expert performance. The time to train a novice can be computed from a transformation of these expert methods to novice level representation if greater accuracy in terms of eliciting or inferring mental operators can be achieved.

The basic research for predicting time to learn and transfer time between related tasks was conducted by Kieras and Bovair (1986) as a result of an approach proposed by Kieras and Polson (1985). Later techniques for quantifying transfer of training were proposed by Singley and Anderson (1989) based upon some of the same assumptions of Kieras and Bovair (1986). The underlying assumption of this basic research was that the amount that must be learned is linearly related to the time it takes to learn. The total time it takes to learn can be determined by the number of new productions which must be learned by the student. Since the amount of time it takes to learn a new production unit is known, counting the number of new production rules which must be learned can predict training time.

The majority of the past research work conducted in this area focused upon a demonstration of this underlying assumption by establishing the time it takes to learn a new production unit and then predicting total training time. An important consequence of this work was the establishment of a number of requirements which such models must meet in order to predict training time. First, as Singley and Anderson (1989) found the lowest level model possible, consistent with the Keystroke Level of analysis associated with GOMS modeling, is required to make such predictions. That is, both primitive mental and physical operators must be described. Secondly, these models, indicative of the way in which experts structure their knowledge, will not suffice to predict novice training time since expert representations and novice representatives differ in terms of how the subgoals get composed. Therefore, expert representations must be transformed into novice level representations of task knowledge. Experts typically have fewer rules, as a result of composition, which represent their task models when compared to novices. Likewise, experts, unlike novices, do not explicitly check for all feedback in the way of prompts from a system on which they have been expertly trained, whereas novices perform such feedback checks as explicit steps in their production rule representations (Bovair, Kieras and Polson, 1990). Novice rules typically consist of only a single action or step whereas experts compose their rules into a sequence of specific actions, as these single actions or steps are repeatedly executed in a specific order, they then become composed, a sign of the development of expertise.

As to the accuracy with which predictions of total training time can be made from knowledge of the number of new production rules which must be acquired, the results of experimentation are quite comparable. Kieras and Bovair (1986) developed an equation from their research which accounted for 76% of the variance in measures of total training time. Their equation estimated total training time to be:

$$\text{Time} = 85.3 \text{ seconds} + 20.3 \text{ seconds} \times (\text{number of new production rules}).$$

This measure of total training time consisted of reading time plus the time spent in practice to learn to accurately perform the procedures of the task. The y-intercept of their equation (i.e., 85.3 seconds) therefore represents a baseline time which indicates how long it would take someone who knows the task procedures to read the instructions and accurately perform the task. That is, nothing new would be learned by such an individual. This baseline time will obviously vary dependent upon the task to be learned. However, the number of new production rules to be learned is a highly reliable predictor of training time.

Another study conducted by Bovair, Kieras and Polson (1988) produced a regression equation which accounted for 88% of the variance in total time to learn. The task utilized in this study consisted of learning a number of text editing procedures. The regression equation found for this study was:

$$\text{Total Training Time} = 683.2 \text{ seconds} + 34.7 \text{ seconds} \times (\text{number of new rules}).$$

For this later study, the most important predictor variable was again the number of new rules. From these basic research experiments, Kieras (1988) has proposed a generalized equation for purposes of estimating training time as follows:

$$\text{Learning time} = (30-60) \text{ minutes} + 30 \text{ seconds} (\text{number of new production rules}).$$

Of critical importance to making such predictions is determining how to compute the number of new productions to be learned. As indicated by the work of Kieras and

Bovair (1986) and Bovair, Kieras and Polson (1988) each rule which must be acquired by the trainee may or may not get the full learning time charge in predicting total training time. This is because what is learned early in a training session may transfer to rules learned later in the session. Consequently, many rules which are part of the model may share similarity with other rules in the model and do not get the full training time charge. One must then determine how to compute this similarity between production rules given novice level models. The heuristic developed by these researchers proposes that there are four possible outcomes for each candidate rule which is to be learned: (1) a rule can be identical to an existing rule and therefore no new learning is needed, (2) a rule can be generalized to an existing rule which requires very little learning, if any, (3) a rule can be subsumed under an existing generalized rule requiring no new learning, and (4) a rule is a new rule and must be added to the rule set being acquired, requiring the full amount of time to learn. Examples of three of these possible outcomes, with the exception of the identical rule case, is presented in Figure 6 a, b and c, as adapted from Bovair, Kieras and Polson (1990) for a text editing skill at a novice level representation.

Consequently, in order to extend the current modeling capability of CAT, processes for conducting identity matches and generalizations to existing rules must be developed. However, the process for determining such generalizations has been explicitly described by Bovair, et al (1988). As described in example 1 of Figure 58 a, the old rule, NoviceDelete.P5, and the new rule, NoviceCopy.P5, are compared to determine if both the verb and the noun in the goal descriptions are different. If they are both different, then the rules are not similar at all and there can be no generalization. However, if they are different on only one of the terms, in this case, the verb term (e.g. DELETE STRING versus COPY STRING) then the rules are similar enough to merit further processing. The remaining statements in the rule are then compared to determine if the verb term is the

only difference in the remaining clauses of the rule. If so, then the verb term is assigned a

Example 1: New Rule Is Generalized With the Existing Rule		
Old Rule: (NoviceDelete.P5)		New Rule: (NoviceCopy.P5)
IF ((GOAL DELETE STRING) (STEP VERIFY DELETE))		IF ((GOAL COPY STRING) (STEP VERIFY COPY))
THEN ((Verify Task DELETE) (Delete STEP VERIFY DELETE) (Add STEP PRESS ACCEPT)))		THEN ((Verify Task COPY) (Delete STEP VERIFY COPY) (Add STEP PRESS ACCEPT)))
Generalized Rule: (NoviceDelete.P5)		
IF ((GOAL ?X STRING) (STEP VERIFY ?X))		
THEN ((Verify Task ?X) (Delete STEP VERIFY ?X) (Add STEP PRESS ACCEPT)))		
Example 2: New Rule Is Subsumed Under an Existing Generalization		
Old Rule: (NoviceDelete.P5)		New Rule: (NoviceMove.P5)
IF ((GOAL ?X STRING (STEP VERIFY ?X))		IF ((GOAL MOVE STRING) (STEP VERIFY MOVE))
THEN ((VerifyTask ?X) (Delete STEP VERIFY ?X) (Add STEP PRESS ACCEPT)))		THEN ((VerifyTask MOVE) (Delete STEP VERIFY MOVE) (Add STEP PRESS ACCEPT)))
Example 3: New Rule Is Different From All Existing Rules		
Old Rule: (NoviceInsert.P2)		New Rule: (NoviceDelete.P2)
IF ((GOAL INSERT STRING) (STEP CHECK-PROMPT INSERT)) (DEVICE USER MESSAGE Insert What))		IF ((GOAL DELETE STRING) (STEP CHECK-PROMPT DELETE)) (DEVICE USER MESSAGE Delete What))
THEN ((Delete STEP CHECK-PROMPT INSERT) (Add STEP LOOKUP MATERIAL)) (Add STEP PRESS ACCEPT)))		THEN ((Delete STEP CHECK-PROMPT DELETE) (Add GOAL SELECT RANGE) (Add STEP SELECT-RANGE DELETE)))

Figure 58 : Examples of rules showing transfer status.  
(Bovalir, Kieras and Polson, 1988)

single variable for each instance of the verb term in each rule. The result is the generalized rule of example 1, Figure 58 a.

To determine if a new rule can be subsumed under an existing generalized rule, the same process as shown in example 2 of Figure 58 b, defined for generalization, is employed. Of the two rules being compared, one is an existing generalized rule and the other is a new rule. Again, the verb and noun goal descriptions are compared indicating a difference in the verb term (e.g., STRING versus MOVE STRING). The remaining clauses of the rule are examined and found only to differ in the verb term description. This being the case, the new rule is subsumed under the existing generalization and therefore not given any learning time charge. If the rules of these examples (e.g. DELETE, COPY, and MOVE) are learned in that order, only the rule DELETE will get the full charge for learning time. The remaining rules will not receive any learning time charge.

The second requirement for making such predictions is getting at the lowest level of analysis as exemplified by the Keystroke Level model of methods, and then converting these methods into a novice representation. One can only make reasonable predictions regarding time to learn when the cognitive models generated represent rules which are characteristic of novice representations. Novice representations differ from expert representations in that each step in a GOMS method is represented as a single separate rule. A step includes not just overt actions like pressing a key, but also covert mental operations such as attending to a prompt on a screen, locating a manual, determining an insertion point on a display, awaiting a verbal response, etc. Novices also explicitly create rules for checking all feedback from a system such that each prompt presented on a screen would be checked and represented as a separate production rule. Experts on the other hand would not describe such checks and may have different numbers of rules in their

representations of a specific method due to differences in the way the rules of a method get composed by different experts. The training time for these different compositions of rules may differ making it difficult to predict training time from this level of representation. Figure 59 a and b demonstrate the differences between a novice representation of a method and that of an expert. As can be seen from Figure 59 b, depicting a novice's method representation, the rule, Start Method, starts execution by adding an assertion to memory to do the first step. Rule, Method Rule 1, then checks to see if the first step has been asserted in memory on its condition side and then executes the step and adds a note that the next step is the finish step. The rule, Finish Method, checks to determine if the steps have been finished and adds the note to memory that the method has been finished. As depicted in Figure 59 a, representing the experts rule for the same method, there are a fewer number of rules. Those rules which were chained together in the novice version of the method have been collapsed into two rules in the expert's representation of the method.

**(a) Example expert method**

```
(SelectMethod      IF      ((GOAL PERFORM <TASK>))
(NOTE <SPECIFIC CONTEXT>)
(NOT (NOTE EXECUTING <TASK>)))
      THEN      ((Add GOAL <DO SPECIFIC METHOD>))
(Add NOTE EXECUTING <SPECIFIC METHOD>)
(<DoAct FirstAct>)
(Add STEP DO <Finish Step>)))
(FinishMethod      IF((GOAL PERFORM <TASK>))
(GOAL <DO SPECIFIC METHOD>)
(STEP DO <Finish Step>)
      THEN((Add NOTE <TASK> PERFORMED)
(Delete GOAL PERFORM <Task>)
(Delete GOAL <DO SPECIFIC METHOD>)
(Delete STEP DO <Finish Step>)
(Delete NOTE EXECUTING <SPECIFIC METHOD>)
(Delete NOTE <SPECIFIC CONTEXT>)))
```

**(b) Example Novice Method**

```
(SelectMethod      IF      ((GOAL PERFORM <TASK>))
(NOTE <SPECIFIC CONTEXT>)
(NOT (NOTE EXECUTING <TASK>)))
      THEN      ((Add GOAL <DO SPECIFIC METHOD>))
(Add NOTE EXECUTING <TASK>)))
(FinishSelect      IF      ((GOAL PERFORM <TASK>))
(NOTE <SPECIFIC METHOD> FINISHED))
      THEN      ((Add NOTE <TASK> PERFORMED)
(Delete GOAL PERFORM <TASK>)
(Delete NOTE <SPECIFIC CONTEXT>)))
(StartMethod      IF      ((GOAL <DO SPECIFIC METHOD>))
(NOT (NOTE EXECUTING <SPECIFIC METHOD>))
      THEN ((Add NOTE EXECUTING <SPECIFIC METHOD>))
(Add STEP DO <First Step>)))
(MethodRule1      IF((GOAL <DO SPECIFIC METHOD>))
(STEP DO <First Step>))
      THEN((<DoAct First Act>)
(Add STEP DO <FINISH STEP>)
(Delete STEP DO <First Step>)))
(FinishMethod      IF((GOAL <DO SPECIFIC METHOD>))
(STEP DO <Finish Step>)
      THEN ((Add NOTE <SPECIFIC METHOD> FINISH)
(Delete GOAL <DO SPECIFIC METHOD>)
(Delete STEP DO <FINISH STEP>)
(Delete NOTE EXECUTING <SPECIFIC METHOD>)))
```

**Figure 59:** Comparison of an expert level representation of a method to a novice level representation of a method.

In view of these requirements for developing models which can be used to predict time to learn, as well as transfer time, the current version of CAT must be augmented to translate method descriptions into production rule forms such that each step of a method is represented as a single production rule.

Translating GOMS model descriptions into a production rule representation would be a tedious task and would impose rather strict constraints on the user. Additionally, numerous routines would need to be developed to transform user inputs into code executable by a production system. The objective of this research was to provide the user with a simple interface that allows for a natural English-like description of the users task. Moreover, for predicting training time, it has been demonstrated by Kieras (1988) that such predictions can be made employing his "Natural GOMS Language"(NGOMSL) notation. NGOMSL is a high level language similar to the English descriptions currently processed by CAT which are isomorphic to production rule representations executable by a production system. Each NGOMSL statement represents a simple production rule and therefore, can be used to make predictions concerning training time and transfer as is the case with production rule representations. These NGOMSL statements, however, are much easier to work with from a computational point of view and are consistent with the current CAT version. Fortunately a number of style rules have been created by Kieras (1988 and 1990) for his NGOMSL notation system which mimics those style rules which are applied to traditional production rule system development.

These NGOMSL style rule guidelines when applied to cognitive task descriptions produce models from which the necessary training time and transfer time predictions can be made, without the need for a strict production rule notation form. Therefore, these style rules which are adapted from those applied to formal production rule notation can be

applied with greater ease to attain the objective of predicting execution time, training time and transfer time.

### **Practical Benefits Relative to Manpower, Personnel and Training**

There are a number of practical benefits which can result from a capability which predicts time to learn. First, in the development of new military systems, the cost to deploy such systems in part is determined by the cost to train individuals to expertly use a system. This cost to train translates into the time required to train an individual to a required level of expertise. Consequently when establishing a budget for system deployment and the life cycle cost for new systems, the cost to train users can be reliably predicted. Second, differences between two alternative designs of a system which accomplish the same mission can be assessed in terms of the difficulty which will be encountered in learning to operate the alternatives. This assessment can be used in the decision process when making a choice between the alternative options. Third, in the conduct of a job analysis, it is important to determine what tasks are most compatible such that they may be grouped together in job descriptions. Furthermore, it is important to determine what personnel with prior task experience can most easily transfer their knowledge between jobs and job tasks. The training time prediction capability which is proposed could be used for such purposes since the analysis to determine training time examines the similarity between knowledge units within a specific task, as well as, between different tasks. The greater the similarity between knowledge units, the greater the transfer and the lesser the training time required to master the task(s). Therefore, the ability to predict training time or learning difficulty has a number of practical benefits for selecting between alternative system designs, establishing the degree of transfer between tasks, planning and budgeting manpower training costs and for job design.

## Acknowledgment

The author wishes to acknowledge Dr. Stanley Collyer and Dr. Susan Chipman of the Office of Naval Research for their support and enthusiasm for this project.

## References:

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R., Kline, P. J. and Beasley, C. M. (1980). Complex learning processes. In R. E. Snow, P. A. Federico, and W. S. Montague (Ed), *Aptitude, learning and instruction: Cognitive process analyses*. Hillsdale, N. J: Lawrence Erlbaum Associates.

Anderson, J. R., Greeno, J. G., Kline, P. J., and Neves, D. M. (1981). Acquisition of Problem Solving Skill. In J. R. Anderson (Ed) *Cognitive Skills and Their Acquisition*. Hillsdale, N. J. Erlbaum.

Anderson, J. R., Conrad, F. G., and Corbett, A. J. (1989). Skill acquisition and the LISP tutor. *Cognitive Science* 13, 476-505.

Bovair, S., Kieras, D. E., and Polson, P. G., (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. *Human Computer Interaction* 5, 1-48.

Card, S., Moran, T. P., and Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Chi, M. T. H., Bassock, M., Lewis, M. W., Reimann, P., and Glaser, R. (1989). Self explanation: How students study and use examples in learning to solve problems. *Cognitive Science* 13, 145-182.

Corbett, A. T., and Anderson, J. R. (1989, May). The LISP intelligent tutoring system: Research in skill acquisition. In *Proceedings of the 4th International Conference in Artificial Intelligence in Education* (pp. 64-72) Amsterdam.

Gray, W. D., John, B. E., and Atwood, M. E., (1992). The process of Project Ernestine or an overview of a validation of GOMS. *Proceedings of CHI'92*. ACM, New York.

John, B. E., (1990). Extensions of GOMS analysis to expert performance requiring perception of dynamic visual and auditory information. *Proceedings of CHI' 1990 ACM*, New York, 107-115.

Kieras, D. E., and Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man - Machine Studies* 22, 355-394.

Kieras, D. E., and Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. *Journal of Memory and Language*, 25, 507-524.

Kieras, D. E., (1988). Towards a practical GOMS model methodology for user interface design. In Hilander (Ed) *Handbook of Human Computer Interaction*. Elsevier: North Holland, 135-157.

Kieras, D. E. (1991). A guide to GOMS task analysis. *Computer-Human Interaction (CAI)'91*, Tutorial.

Kieras, D. E., and Meyer, D. E. (1992). EPIC: A human information-processing architecture with application to multiple-task performance. *Computer-Human Interaction (CHI)* September.

Klahr, D., Langley, P., and Neches, R. (1987). *Production Systems Models of Learning and Development*. Cambridge, MA: The MIT Press.

Langley, P., Simon, H. A., Bradshaw, G. L., and Zytkow, J. M. (1987) *Scientific Discovery: Computational explanations of the creative process*. Cambridge, MA: The MIT Press.

Miyake, N. (1986). Constructive interactive and the interactive process of understanding. *Cognitive Science*, 10, 151-177.

Neves, D. M., and Anderson, J. R. (1991). Knowledge compilation: Mechanisms for the automation of cognitive skills. In J. R. Anderson (Ed) *Cognitive Skills and Their Acquisition*. Hillsdale, N. J: Erlbaum.

Newell, A., and Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, N.J: Prentice-Hall.

Newell, A. (1973). Production Systems: Models of control structures. In W. G. Chase (Ed). *Visual Information Processing*. New York: Academic Press, 463-562.

Peck, V. A., and John, B. E. (1992). Browser-Soar: A computational model of a highly interactive task. In P. Bauersfeld, J. Bennett and G. Lynch (Eds) CHI'92 Conference Proceedings, ACM, New York.

Polson, P. G. (1986). A quantitative theory of human-computer interaction. In Carroll (Ed) *Interfacing Thought: Cognitive Aspects of Human Computer Interaction*. Cambridge, MA: The MIT Press.

Qin, Y., and Simon, H. A. (1990). Laboratory replication of scientific discovery processes. *Cognitive Science*, 14, 281-312.

Simon, H. A. (1978). On the forms of mental representation. In C. W. Savage (Ed), *Perception and Cognition: Issues in the Foundations of Psychology*. Minneapolis: University of Minnesota Press.

Singley, K., and Anderson, J. R. (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard Press.

Thagard, P. (1992). Adversarial problem solving: Modeling an opponent. *Cognitive Science* 16, 123-151.

VanLehn, K., Jones, R. M., and Chi, M.T.H. (1991). Modeling the self explanation effect with Cascade 3. In Hammond and Gentner (Ed). *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. Hillsdale, N. J: Erlbaum.

Williams, K. E., Reynolds, R. E., Carolan, T. F., Anglin, P. D. and Shrestha, L. B. (1990). An evaluation of a methodology for cognitively structuring and adaptively sequencing exercise content for embedded training. TR89-035, Cognitive Science Program, Office of Naval Research, Arlington, Virginia.

Williams, K. E. and Reynolds, R. E. (1991). The acquisition of cognitive simulation models: A knowledge-based training approach. In Fishwick and Modjeski (Ed), *Knowledge-Based Simulation: Methodology and Application*. New York: Springer-Verlag.

Williams, K. E. and Kotnour, T. (1993). *Knowledge Acquisition: A review of manual, machine aided and machine learning methodologies*. Office of Naval Research Interim Technical Report, Contract #N00014-91-J-5-1500.

## REPORT DISTRIBUTION LIST

Dr. John R. Anderson  
Department of Psychology  
Carnegie Mellon University  
Schenley Park  
Pittsburgh, PA 15213

Dr. William J. Clancey  
Institute for Research on Learning  
2550 Hanover Street  
Palo Alto, CA 94304

Dr. Albert T. Corbett  
Department of Psychology  
Carnegie-Mellon University  
Pittsburgh, PA 15213

Dr. Michael Drillings  
Basic Research Office  
Army Research Institute  
5001 Eisenhower Avenue  
Alexandra, VA 22333-5600

Dr. Helen Gigley  
Naval Research Lab Code 5530  
4555 Overlook Avenue, SW  
Washington, DC 20375-5000

Dr. Robert Glaser  
LRDC University of Pittsburgh  
3939 Ohara Street  
Pittsburgh, PA 15260

Professor Joseph Goguen  
PRG University of Oxford  
11 Keble Road  
Oxford OX13QD  
United Kingdom

Dr. Sherrie Gott  
AFHRLMOMJ  
Brooks AFB, TX 78235-5601

Dr. T. Govindaraj  
GA Institute of Tech  
School of Industrial & Systems Engineering  
Atlanta, GA 30332-0205

Dr. Marilyn K. Gowing  
Office of Personnel R&D  
1900 E St., NW Room 6462  
Office of Personnel Management

Washington, DC 20415

Dr. James G. Greeno  
School of Education  
Stanford University  
Room 311  
Stanford, CA 94305

Dr. Barbara Hayes-Roth  
Knowledge Systems Lab  
Stanford University  
701 Welch Road, Bldg. C  
Palo Alto, CA 94304

Dr. Edgar M. Johnson  
Technical Director  
U.S. Army Research Institute  
5001 Eisenhower Avenue  
Alexandria, VA 22333-5600

Dr. David Kieras  
Technical Communication Program  
Tidal Bldg., 2360 Bonisteel Blvd.  
University of Michigan  
Ann Arbor, MI 48109-2108

Dr. Walter Kintsch  
Department of Psychology  
University of Colorado  
Boulder, CO 80309-0345

Dr. Susan S. Kirschenbaum  
Code 2212, Bldg. 11711  
Naval Underwater Systems Center  
Newport, RI 02841

Dr. Janet L. Kolodner  
GA Institute of Tech  
College of Computing  
Atlanta, GA 30332-0280

Dr. Ryszard S. Michalski  
Center for Artificial Intelligence  
George Mason University  
Science and Tech II, Room 411  
4400 University Drive  
Fairfax, VA 22030-4444

Dr. Andrew R. Molnar  
Applications of Advanced Technologies, Rm. 635  
National Science Foundation  
Washington, DC 20550

Dr. Ben B. Morgan, Jr.  
Department of Psychology  
University of Central Florida  
Orlando, FL 32816-0150

Dr. Stellan Ohlsson  
Learning R&D Center  
University of Pittsburgh  
Pittsburgh, PA 15260

Institute for the Learning Sciences  
Northwestern University  
1890 Maple Avenue  
Evanston, IL 60201

Dr. Peter Pirolli  
School of Education  
University of California  
Berkeley, CA 94720

Dr. Martha Polson  
Department of Psychology  
University of Colorado  
Boulder, CO 80309-0344

Dr. Peter Polson  
University of Colorado  
Department of Psychology  
Boulder, CO 80309-0344

Dr. Fred Reif  
CDEC Smith Hall  
Carnegie-Mellon University  
Pittsburg, PA 15213

Dr. Charles M. Reigeluth  
Chairman, Instructional Systems Technology  
School of Education, Rm. 210  
Indiana University  
Bloomington, IN 47405

Dr. Gilbert Ricard  
Mail Stop K01-14  
Grumman Aircraft Systems  
Bethpage, NY 11714

Mr. W. A. Rizzo  
Head, Human Factors Division  
Naval Training Systems Center  
Code 26  
12350 Research Parkway

Orlando, FL 32826-3224

Dr. Michael G. Shafto  
NASA Ames Research Center  
Mail Stop 262-1  
Moffett Field, CA 94035-1000

Dr. Derek Sleeman  
Computing Science Department  
The University  
Aberdeen AB9 2FX  
Scotland United Kingdom

David Smith  
NRAD Building 54  
2717 Catalina Blvd.  
San Diego, CA 92152-5000

Dr. Douglas Towne  
Behavioral Technology Labs  
University of Southern California  
1228 Spring Street  
St. Helena, CA 94574

Dr. Wallace Wulfeck III  
Navy Personnel R&D Center  
Code 51  
San Diego, CA 92152-6800